

METHOD AND APPARATUS FOR DISPLAYING  
DATA STORED IN LINKED NODES

This application claims the benefit of U.S.  
Provisional Application No. 60/135,740, filed on  
May 25, 1999.

FIELD OF THE INVENTION

The invention is a computer implemented method  
of storing, manipulating, accessing, and displaying  
data and its relationships, and a computer system  
(with memory containing data) programmed to implement  
such method.

BACKGROUND OF THE INVENTION

Today's GUI (graphical user interface) software  
requires a great deal of training for proficiency.  
They rarely show their context and rarely show  
internal data and control points. The actions of  
applications should be fast and easy to use, not  
confusing and restrictive.

Shortcomings of current technologies include the  
following:

*Awkward Navigation:* Navigating a current GUI is  
to learn how the programmers have organized the data  
and the control points. Today's UI (user interface)  
relies on windows and pull-down menus which hide  
prior ones as the user makes selections, and an  
often-complex series of mouse selections to navigate  
to the desired control point. In performing an  
action after traversing a path of pull-down menus and  
button selections on different sub-windows, the user  
may be returned to the beginning, forced to retrace  
their steps if they wish to apply an action  
repeatedly. The UI does not shift from one state to  
another in a smooth way.

## BEST AVAILABLE COPY

PATENT

*Relationships are Invisible:* Current GUI's with pop-up menus and layers of text-based windows.

Today's GUI's suffer from a list mentality: rows of icons with little or no meaning in their position, no relationships readily visible between themselves or to concepts which the user may be interested in or have knowledge of.

The standard GUI concept of a canvas with buttons and pull-downs require reading and thinking about the meaning of text to, hopefully, grasp the model of control flow. Virtually no information is given outside of static text to help the user understand relationships and control points.

*Access to Crucial Data is Confusing:* Gaining access to fields or parameters in current applications can also be a challenge. For example, the Microsoft POP mail retrieval service uses a configuration profile defined by the user to access the mail server, and displays the name of that configuration when logging. Although one sees evidence of the existence of this configuration, how does one change it? The answer may not be obvious. An object-oriented system that implemented all actions on an object could also provide the mechanism of self-deletion, but this helps only if the object itself is visible and accessible via the GUI. This is what DataSea does. Windows technology is moving in this direction, by having many words in menus modifiable, but DataSea achieves this automatically by virtue of its design.

*GUI Design is bug-prone:* A complex GUI, such as a presentation and control system for database administration, today consists of many canvases and widgets which hopefully present, through multiple

mouse clicks, all information that is available. Any data can be changed by any part of the program, and this leads to bugs if the programmer can not easily see these interactions. A DataSea presentation of data and control shows all the objects and their relationships, and thus shows immediately what nodes can affect changes to the data, reducing bugs. To turn a DataSea view into an "application" means to set internal parameters, create and link together application nodes, and add programmatic instructions to these nodes. DataSea will implement a means to invoke these instructions.

*Interoperability Conflicts:* DataSea can serve as the single source of data for any application. Any RDBMS (relational database management system) can do this, but DataSea is completely flexible in its data storage and linkage, guaranteeing forward compatibility by eliminating the risk of changes to database structure and entity-relationships of RDBs (Relational Databases).

Two separate DataSea databases can be joined, and automatic linkage routines will merge them without programmer effort. This is generally impossible in RDBs. This joining can occur by simply adding nodes and links from the two data sets, and adding together the contents of the master index, NameListObj. Or, the two data sets can be blended, merging their contents: taking two nodes with the same name from the two separate data sets, creating one node which has all the links from the two separate nodes.

In most storage systems, especially RDBMS's, the user must know how information is stored in the computer and which parameter or parameters that the

computer is using to store the data, as well as the proper range of values. This is often non-intuitive and may seem somewhat arbitrary for a non-technical user. Ideally, the computer would better mimic human associative memory, allowing the user to look for new information associated with that which is better known, such as a particular context, or a range of values without regard to parameterization, to specify the target of interest.

OLAP (online analytical processing) and data mining require both analytical models and custom code to apply these models to particular database structures. These customizations may be hard-coded, non-portable and irrelevant to the model. The DataSea API (application programming interface) provides access to all data while eliminating the need to worry about database structure such as tables, columns and foreign keys.

Shortcomings of more recent data presentation technologies include the following:

*Non-linear Viewing:* Up-coming non-linear presentation tools such as fish-eye or hyperbolic views do not address the difficult problem of how to lay out the data and their relationships before the viewing method is applied. These may be useful, but do not address the difficult issue of how the graph is laid out initially. Nor are they appropriate for highly linked data sets, because the plethora of links resembles a cobweb from a psychotic spider.

*Virtual Reality:* VR takes advantage of visual clues and spatial awareness, but only for data sets that may be appropriately mapped to a 3-dimensional space. Generally data is N-dimensional and thus, in

general, virtual-reality which models information as physical objects in 3D space is inappropriate for viewing arbitrary data.

5       Voice Interface: even more than a GUI, voice control needs a smooth transition from state to state in response to commands so that the user can follow what is happening. GUIs hide previous states with new windows, while the present invention moves objects gradually and continuously in response to  
10       programmatic or user events.

SUMMARY OF THE INVENTION

      The inventive method (referred to as "DataSea") is a method for storing, accessing, and visualizing information. Data is stored into nodes and  
15       visualized as a 'sea' of linked nodes. Nodes can contain anything such as documents, movies, telephone numbers, applications or words containing concepts. Interactions with the user organize the data from a defined, and then refined, point of view, with  
20       relevant data brought to their attention by smooth changes in the data's appearance.

      Essentially, a handful of nodes (which are typically selected by value) are linked to the point of view turning the web of data into a hierarchical  
25       network. Further order is imposed by the use of two types of commands: one which relies on the data values, the other on the links and types of the data nodes.

      The user typically enters words into a computer  
30       programmed in accordance with DataSea and watches DataSea's response. Individual nodes are rendered according to the sequence of nodes between themselves and the point of view, allowing different

presentations of data. Applications may be stored into nodes and can be located and executed by DataSea. These applications can operate on and present information from DataSea in the conventional manner, with windows and menus, or by using the DataSea mechanisms of visualization, including the so-called virtual reality mode ("VR-mode") which supports deterministic data placement as needed in such things as forms and spread sheets.

Examples of use include:

1. Entering keywords and phrases selectively retrieves and emphasizes different data types such as loose notes and email, but these entered keywords and phrases need not match exactly the content of the resulting emphasized data.
2. Direct and specific information is retrieved: The user enters the name of "Jim Smith" followed by an appropriate command such as "back" or "and" along with the phrase "Phone number" which refers to a pre-existing AN. Jim's telephone number becomes obvious as it approaches the point of view, which in this case is "Jim Smith".
3. A manufacturing facility has test data from machines and 3-D models of those machines. These data sources are integrated and the user can visualize the facility from different points of view, e.g. a virtual reality mode, tabular presentation or the standard DataSea network connectivity display.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a diagram of relationships between nodes which can be visualized in accordance with the invention.

5 Fig. 2 is a flow chart of steps performed in accordance with the invention.

Fig. 3 is a block diagram of an embodiment of the invention.

10 Each of Figs. 4, 5, 6, 7, 8, and 9 is an example of a display generated in accordance with the invention.

PREFERRED EMBODIMENT OF THE INVENTION

15 This is a disclosure of computer implemented methods for storing, manipulating, accessing and displaying data and its relationships, and of computer systems programmed to implement such methods.

20 These methods are flexible, applicable to any kind of data and are useful to diverse groups of users. A prototype is implemented in the Java programming language.

25 Data is stored into nodes which are linked together. All nodes contain variables, including descriptions, types, magnitudes and timestamps. Links also contain information about themselves, including connection strength of the link and descriptive information.

30 Data is accessed and modified based on the values of data, their relationships, the values of DataSea parameters and links between nodes, rather than pre-determined locations in memory, as is done in most programming models.

PATENT

Any existing application can be emulated in DataSea by creating and linking appropriate nodes. Positions of nodes as displayed on the screen are a result of processing force parameters rather than pre-determined positions.

Commands to DataSea are not chosen from hierarchical lists or menus, but rather they are

1. Simple navigation commands;
2. Words (data values) generated by the user; and
3. References to nodes brought to user's attention.

This approach to the command interface, and the smooth changes of state in visual feedback, lends DataSea to voice input and thus wireless 'PDA' (personal digital assistant) type devices.

Key aspects of the invention are:

1. nodes which can contain any type of data, links between the nodes;
2. manipulation of internal parameters of nodes and links;
3. visualization of nodes and their internal parameters;
4. smooth transitions during changes in state of the network;
5. integration of virtual reality representations;
6. simple commands and low learning curve;
7. increased robustness to imprecise commands as the data set grows and matures;
8. obviation of the need to predetermine data structures when entering new data; and
9. integration of legacy data with the structure inherent in it.

DEFINITIONS:

- 'POV' stands for 'Point of View' and is the designation given a node from which many operations are begun, such as defining a hierarchy of distance in terms of links of any node to the POV.
- 'DN' refers to a data node containing specific values of a parameter. 'AN' refers to an abstract node, which contains a summary, abstract, or explanation of the data stored in one or more DNs, and represents a concept or parameter name which can link and thereby group one or more DNs. E.g., "address" is an AN, while the specific address "123 Main St." is a DN.
- DN("abc") refers to a data node named "abc". AN("xyz") refers to a AN named "xyz".
- 'magnify' means to change the value of the magnitude variable ('mag') or related variables inside a node and or its links.
- 'distance' refers to the minimum number of links between two nodes, also called the link-distance.
- 'conceptual distance' refers to a more complex function of link distance and other parameters including mag, link-connection-strength and total number of links to a node. For instance, the conceptual distance between two nodes which are several links apart can be proportional to the product of the link connection-strengths times the product of the mag's divided by a function of the link-distance.
- 'near' refers to a relatively low distance or conceptual distance.

- 'commands' in DataSea are any programmatic methods of accessing, manipulating, creating or deleting data structures or elements of DataSea.
- 5     • 'applications' in DataSea are programs which use or modify data, links or resources of DataSea, and modify internal parameters of nodes or links, and or create or delete nodes.
- 10    • 'environment checking' refers to getting information about nodes and links in the neighborhood of or nearby one or more nodes.
- 'abbreviated' refers to incompletely rendering and positioning nodes.
- 'distal path' is a sequence of linked nodes, each node distal to the prior one.
- 15    • 'multiple paths' refers to more than one route, through links, between two nodes.
- 'interaction with DataSea' refers to either user interaction or programmatic interaction. User interaction is typically through the use of
- 20    DataSea query language (keyboard or voice interface) and mechanical means such as a mouse. Programmatic interaction can result from DataSea nodes themselves or external programs.
- 25    • 'distal' refers to those nodes having a higher link-distance value (node.dist) than the link-distance value (this.dist) of the reference point or 'this': that is, node.dist > this.dist.

30    Proximal is the opposite, referring to lower distance.

- 'primary node' refers to a node directly connected to the reference point or 'this'.
- 'child' is a primary distal node, while 'parent' is a primary proximal node.

- 'setting a POV' means assigning an existing node as a point of view, or creating a new node and linking it directly to a number of existing nodes, specified at the time of creation of the POV or later.
  - "spreading mode" refers to the rules used in applying algorithms, possibly recursively, from one node to its neighbor or neighbors. Criteria might be dependent on proximal or distal progression, mag, CS (Connection Strength of a link), potentiation level, or other factors.
  - 'Aliasing' refers to the mapping of a node to other nodes or a range of nodes. For instance, when an action is performed on the "tomorrow" node, it executes enough code to link itself to a node having the (absolute) value of tomorrow's date, dropping links to any other dates previously established. Linking a node to "today" results in linking to the node representing the absolute date. Depending on the spreading mode, related times will be more or less affected by commands. Nodes with TimeStamps closer to tomorrow will be affected more than those further away. Spreading can also be dependent on values and a range, e.g. decreasing the effect of "more" as  $1/((\text{current\_value} - \text{optimal\_value})/\text{fuzziness\_range})$ . Integration of a GPS receiver, or simply assigning a static location in a map, will be used to alias the word "here".
- "Presented data set" is what the user sees at the moment, being a collection of nodes with (mag > some threshold).

"Target data set" is an ideal collection of nodes representing the information the user is searching for.

5        "Traveling distally" means going from node A to node B only if B.dist is > A.dist.

      "Traveling proximally" means going from node A to node B only if B.dist is < A.dist.

10        "Traveling downstream" means going from node A to node B only if A.getPol(B) is '-'. The polarization of the link between A and B can be set on entering the link between A and B, the order of A and B determining the polarization:

      A.link(B, {polarization=}true)  
      means

15        A.getPol(B) is '-'. Typically, nodes A and B are linked such that node B is downstream from node A when node A is more general than node B (so that by traveling downstream from node A to Node B, a user encounters more specific, rather than less specific, information). In characterizing a node, a user  
20        usually wants to see progressively more general (broader) information about it.

      "Traveling upstream" means going from node A to node B only if A.getPol(B) is '+'.  
25        A "Recently visited" node denotes a node being used in an operation which traverses the network. The node's TimeStamp is updated on a visit.

      A "Recently visited" node denotes a node being used in an operation which traverses the network. The node's TimeStamp is updated on a visit.

30        As shown in Fig. 1, each node has a link to at least one other node. Each link is defined by three values: CS (which is Connection Strength of a link, initially set to 1.0), Description (which is a free-form String describing the node), and Type (For example DN (data node) and AN (abstract node)).

Fig. 2 is a flow chart of steps performed in accordance with the invention. First, a Point of View (POV) is set. Then, internal parameters associated with the POV are set. The internal  
5 parameters include those listed (with "pressure" denoting the values of summed forces influencing a node's position, "forces" denoting parameters indicative of pushing or pulling of the node's position relative to other nodes: positive or  
10 negative values calculated by repetitive interactions between any node and others, and global forces such as a 'drift' or 'gravity' biasing the positions of nodes, and "positions" denoting the location on the computer display, or more generally the doublet or  
15 triplet representing a node inside a virtual space, viewed on the computer's display). Then, feedback is provided to the user by displaying representations of the nodes on a display device in accordance with the internal parameters set in the second step.  
20 Optionally, auditory feedback is also provided to the user during the third step. As a fourth step, the user interacts by providing commands for modifying the internal parameters (both node parameters, such as magnitude, and link parameters, such as distance)  
25 set in the second step. The commands can include any of those listed, or any of those discussed below. In response to a command specified in the fourth step, the second step is performed again (to reset the internal parameters) and then the third step is  
30 performed again (to provide feedback indicative of the reset parameters).

EXAMPLE OF USING THE INVENTION:

5 A candy factory supervisor performs the following tasks (there are many different commands and choices of values which will give similar results):

- 10 • views the facility (the candy factory) to get an overview of its status and sees a VR (virtual reality) rendering of the various stations, color-coded for activity (by using the command "show factory");
- 15 • reviews the recent temperature history of one of the problem machines, a chocolate melter and sees the machine with clusters of data nearby ("reset, show Choc\_2\_Melter and Temperature and TimeLine and recent");
- 20 • views the melter and groups data from it and sees the abstract nodes "Temp", "Up-Time" and "Operators" which categorize the data ("reset, show Choc\_2\_Melter, group")
- 25 • checks for other applications which use data from the chocolate melter ("apps") and after seeing the familiar application named "AppMelterGraphs" invokes this canned application showing graphs of the melter's history ("reset, show AppMelterGraphs and Choc\_2\_Melter");

PATENT

- saves this point of view named "Daily" ("save Daily"). Later in the week, after forgetting its name is reminded by asking for saved POV's from last week ("show Saved and LastWeek");
- 5 • checks unread mail ("reset, show Mail and unread and TimeLine"), and now de-emphasizes replies by himself ("less Me")  
  
    'Me' is an alias node, linking directly to a node representing his user ID;
- 10 • begins an email to co-worker ("reset, show 'John Smith', AppMail"). "AppMail" is a canned application which starts from the current point of view and searches the neighborhood for two data nodes: a data node directly connected to an abstract node that is equivalent to "Name", and another data node connected to the abstract node "Address". It then formats a text window for composing an email message; and
- 15 • enters appointment with Scharffen Berger chocolate supplier (input "mtg tomorrow 4pm 'Scharffen Berger'") and sees TimeLine with the event for April 28, 1999, more info about the contact person at Scharffen Berger is visible.
- 20

25 DataSea is a comprehensive program that stores, manipulates and visualizes all forms of data. Visually animated objects, or nodes, represent data or abstract concepts. Interactive commands (which

something like verbs) operate on nodes and the links between them (which act something like nouns). These commands change internal parameters of the nodes and links. These parameters are visualized by qualities such as position and size. Certain nodes are emphasized, presenting information. The user finds the data or resource needed without knowledge of the data structure.

Unusual features of DataSea include relatively natural commands, robustness to imprecise queries, ability to generalize, absence of restrictive structure, use of semantic information and smooth transitions between visual states of the user interface. The simple commands and feedback from smooth visual transitions is key in integration DataSea with a voice interface.

The front-end of DataSea is a query interpreter and visualization system, and on the back-end is a database and API (application programming interface). Briefly, one sees a 'Sea of Data', and after each of a series of commands, one sees increasingly relevant data more prominently.

DataSea nodes act something like nouns of a natural language, and DataSea commands something like verbs. Here are the principal steps involved in a user query:

- Establish a point of view (either an existing node or a new, 'blank' one: if new, enter one or more reasonable values for broad, relevant terms of the query).

- Invoke commands (such as 'show', 'back', 'similar', 'abstractions') followed by more words of the query.
- Directly (e.g. 'more wordXXX') or indirectly (via commands such as 'group', 'similar', etc.) manipulate the presentation, progressively emphasizing information that is more relevant.

5  
10 In a preferred embodiment, DataSea is a pure-Java application that can serve in a range of roles. It can view and control existing and legacy data such as email, documents, file directories and system utilities. It can ultimately serve as the principal UI to a system managing all data and system resources of a personal computer or workstation.

15 The natural ability of people to recognize visual patterns can be leveraged to convey information rapidly to the user. For instance, certain algorithms which depend on particular node and link configurations can render and position those nodes for rapid recognition. For example, if a  
20 target DN is surrounded by intermediate DNs which are themselves linked each to a distal AN, then those intermediate DNs are probably describing the target DN. The number of intermediates is then a measure of  
25 how much information is known about the target DN.

Its ability to gracefully reduce the complexity of the visual output means that a wireless hand-held

client can be used to quickly browse and retrieve information from a remote server.

5       The simplicity of commands and accessibility of DataSea to the novice user lends itself to voice commands that can be used to navigate and control the display of DataSea.

10       The simplicity of DataSea's data structure allows easy acquisition and integration of legacy data into DataSea. Because new data is integrated with old, the acquisition of new data not only allows its retrieval by the user, but also enhances the user's retrieval of older data. Thus, as DataSea matures in its data content, queries are more robust to imprecise terms from the user. Since DataSea captures the information in the data and its structure from legacy databases, applications in DataSea can emulate legacy applications, while of course making this information available to broader use within DataSea.

20       While DataSea can emulate a RDBMS, without the complications of tables and foreign keys, the rich connections of DataSea and its ability to insert abstract nodes opens the way for neural-type processing. Learning-by-example is one example of that new capability. Learning-by-example refers to adjusting mag and CS values by 'voting' (via 'more' or 'less', for example) on DNs, without relying on ANs. This selects DNs which the user especially likes or dislikes. Applying commands to DNs (such as files or URLs) changes not only the mag of each DN, but changes the CS and mag in its neighborhood,

5 typically spreading through related ANs, thereby  
changing the mag of other DNS in the neighborhood,  
i.e., having similar qualities as the DNS that the  
user liked. A different point of view applied to  
DataSea, by virtue of different connections and  
connection strengths, changes the presentation of  
data as fundamentally as changing the database design  
in a relational database, but much more easily.

10 DataSea can be used to perform simple web-  
history viewing, data mining, and can be used as the  
principal Desktop UI for a computer showing all of  
the computer resources.

15 *LEGACY DATA AND NETWORK TOOLS:* Viewing domains  
such as file systems, web history or HTML documents  
and computer networks are obvious uses of DataSea and  
are early targets of DataSea. Applied to a web  
browser, the text of links to the current URL can be  
retrieved and parsed into DataSea, in effect pre-  
digesting it for the user.

20 *VOICE INTERFACE TO WIRELESS HANDHELD DEVICES:*  
The GUI (Graphical User Interface) of DataSea is  
important, but the underlying structure of DataSea  
queries and input methods are curiously appropriate  
for voice and natural language interfacing. Since  
25 queries, input and control of DataSea rely on simple

## PATENT

words DataSea, current voice recognition software can be used instead of text input, and would significantly improve the uniqueness and general usability of DataSea. No other UI uses voice or is as appropriate for voice control. Since the results of many queries may be a short answer, voice generation is an appropriate output method, in addition to or instead of graphic output. For instance, the query 'show John Smith and address' is precise enough to generate one value significantly stronger than others, and therefore amenable to a programmatic decision for selecting which results to submit to voice output. In this way, voice can be a complete communications method, opening the door to remote access via telephone or wireless device.

*PORTAL:* Another opportunity involves selling server time for web searches, giving away client software initially. A typical interaction might involve throwing a number of search terms, asking for a display of abstraction categories or examples of URL's followed by the user judging prominent nodes, repeating as the search narrows.

*DATA WAREHOUSING:* DataSea's data structure and tools lend themselves naturally to data warehousing and mining, each with an estimated worldwide budget in 1999 of nearly \$2 billion. DataSea intrinsically provides data mining and warehouse support. DataSea supports any type of data without specifying in

## PATENT

advance the fields or tables to use. This is good  
for arbitrary user input, such as free-form notes, or  
machine-generated input, such as received data from  
automated test-equipment. DataSea therefore is a  
completely flexible data warehouse.

5

*DATA MINING:* Data mining is supported by  
DataSea's ability to reorganize any data based on  
user-defined point of views, the ability to link any  
and all data, and the ability to store the processing  
of data and applications into DataSea itself.

10

*PRINCIPAL UI:* DataSea can serve as the Desktop  
screen, the principal interface to all system  
services; independent of operating system. It can do  
this on demand, without locking the user into a  
particular operating system.

15

## ARCHITECTURE OF DATASEA

### Java Objects

The most used variables of object Node are Name,  
dist, mag and links[];.

20

Definitions:

PATENT

global variables: Node PointOfView\_node, lastNode;

Class Node extends Object { // Important variables in  
each node

5       Object Data; // contains any computer  
representation of data, and includes get and set  
methods

int dist; // the number of links from this node  
to the POV or another node.

10       int tdist; // a temporary version of 'dist' used  
in calculating the minimum number of links from  
a node to other nodes.

double mag, x,y,z; // x is the 'x' position in  
the DataSea coordinate system

15       double px, py, pz; // px is pressure in x  
direction resulting from positioning routines

double potentiation; // used to make node more  
sensitive to effects such as magnify

20       TimeStamp potentiationTS; // time of last  
potentiation, used to degrade effect of  
potentiation as time passes

String Desc, Type; // used to describe the node.  
Type is typically DN, AN, Event

PATENT

```
LinkObj links[];
```

```
}
```

```
Class DataObj extends Object { // contains any  
computer representation of data, and includes get and  
set Methods
```

5

```
String s;
```

```
...
```

```
getDataAsString().
```

```
...
```

10

```
}
```

```
Class LinkObj extends Object {
```

```
Node linked_node;
```

```
Double CS;
```

```
TimeStamp TS;
```

15

```
String Desc, Type; // used to describe the link,  
may refer to the source of the link, whether its  
an alias or not. usually set by the creator of  
the link.
```

```
}
```

PATENT

Class VRObj extends Object { // VR stands for Virtual Reality

5           double VRx, VRy, VRz; // relative positions in VR space, typically positions offsets from another node identified by recursive calling sequence or information contained in this or in related nodes or links.

          boolean VRlocal, VRenabled;

10           VRShape Shape; // data and methods to render semi-realistically, for Virtual Reality presentation.

          }

Class NameListObj extends Object {

          // Acts as an index for all nodes.

15           // Vector, hashtable or other implementation of all nodes for rapid access based on name and or other fields such as TimeStamp and Desc

          // METHODS

          Node getNodeNamed(String s) {};

20           }

EXAMPLES OF SUBROUTINES USED IN PREFERRED EMBODIMENT

- Node.getChildCount() // return the count of distal links
- 5 • Node.getChild(int i) // returns i<sup>th</sup> link with distance > Node's distance
- Node.getParent()
- Node.getNodeNamed(String s) // finds a node named 's' anywhere
- 10 • Node.getNearbyNodeNamed(String s, int max\_distance, String type) // finds a node named 's' within 'max\_distance' links of Node, having Type 'type'.
- 15 • Node.getConceptualDistanceTo (String s, int max\_distance, String type) // returns result which is a function of distance to the target\_node named 's', Type 'type' and the CS's to and including the target\_node, and the mag of target\_node.
- 20 • Node.get/setNodeLinkedToAN(String an\_name, Data data\_value) get or set the value of the node between 'this' and AN(name)
- 25 • set\_dist(Node starting\_node) // recurses, calculates and sets Node.dist by finding the shortest route to each node by recursing from starting\_node

- `set_POV(Node target_node) //`  
`{ set_dist(target_node); POV=target_node; }`
- `show (String name) { create_POV();`  
`POV.link(getNodeNamed(string)); set_dist(POV); }`

5     EXAMPLES OF USER COMMANDS

Most methods have three versions of arguments:  
( ), (String s), and (Node n1, Node n2 ...). If null,  
then lastNode is used, if String, then matching nodes  
are looked for: both pass one or more nodes to the  
10     third version which takes explicit Nodes.

- `Show(), (Node target) //` link target to point of  
view, create point of view if necessary
- `Abs() (Node target) //` magnify distal ANs  
15     showing category of target (the AN is in a sense  
a category). An AN related to the target by two  
or more intermediate ANs will accumulate  
magnification via those intermediates. Follow  
distal paths, magnifying ANs along the way. Any  
AN along multiple distal paths will be magnified  
20     multiple times. Thus higher level ANs are  
emphasized.
- `Back(Node target) //` working proximally from  
target, increase mag of all until point of view  
is reached

- `And(Node target_1, target_2) // potentiate neighborhood of target_1, then raise mag in neighborhood of target_2 if. potentiated`
- 5 • `More(Node target) // raise mag in neighborhood of target, reducing the amount of change in mag as a function of spread_mode: e.g. proportional to the distance or a constant up to some threshold distance.`
- 10 • `Potentiate(Node target) // similar to More(), but the value of the variable potentiated is increased rather than the variable mag, and the potentiation TimeStamp ('potentiationTS') is updated and used to tell other routines when this was last potentiated. Typically other`  
15 `routines will reduce their modifications to variables as the elapsed time (currentTimeStamp-potentiationTS) increases.`
- 20 • `Sim(Node target) // indicates DNs which are similar to target based on their connections to ANs or other nodes. Similar to abs() but DNs on multiple paths are emphasized. Note: 'abs' and 'sim' use similar mechanisms traversing nodes. One emphasizes ANs resulting in abstracting the categories of the starting point, and the other`  
25 `emphasizes DNs thereby showing nodes that are similar to the starting point.`
- 30 • `Group(Node start, int target_level) { // group DNs around ANs which characterize them. From point of view, go distally until child.dist==target_level. If child is an AN,`

5       then force parent.X = child.X which clusters the  
data nodes between the start and the child  
abstract node onto the abstract node. Wait a  
second or so, letting the data nodes spread  
apart some, and repeat for (target\_level--).

- **Recent()** // magnify nodes with recent  
TimeStamps.

The usage "Recent 1 hour" sets the value "1 hr"  
into the DN between DN(now) and AN(range).

10       where "range x = y" means:

```
(DN(now).getDNhavingANnamed(range)).setData  
("1 hour")
```

15       We next describe some of the above-mentioned  
commands in another way, and we describe other  
commands:

#### DATA MANIPULATION COMMANDS

- **Show**   links specific keyword to a point of  
view, zooms on it and emphasizes it and its  
neighbors.
- 20   •       **Group**   Starting from the point of view,  
secondary (2<sup>0</sup>) data nodes spread distal magnify  
to directly connected abstract nodes, and set

the secondary nodes position next to the largest directly connected abstract node, thereby grouping them.

- 5       •   **Link | unlink**   links specific nodes to a point of view or other nodes.
- **More | Less**   emphasizes specific keywords given by the user and their immediate neighbors.
- 10     •   **Abs(tractions)**   emphasizes abstract nodes related to a data node, higher levels of abstractions being dominant initially.
- **Sim(ilarities)**   emphasizes data nodes which are similar to a selected data node.
- 15     •   **AND**   emphasizes nodes near two or more selected nodes, similar to the boolean 'and' function, although as with most aspects of DataSea, the result is not a binary decision. The non-linearity of the AND operation is adjustable, bringing in more or less of the neighbors. A highly non-linear mechanism akin to neural  
20       'potentiation' can also be used, which can give very precise selectivity to the process of adjusting connection strengths and magnitudes.
- **SS**   Spreadsheet simulation, given one data node, this presents related data nodes in  
25       tabular form with their principal abstract nodes as column-headers. Useful for tabular output.

## PATENT

- **TL** A fast synonym for "zoom TimeLine", "more now". 'Now' is a node updated automatically with the current time, linked to the TimeLine and nodes containing preferences for concepts such as 'recent'.

5

## VIEWING COMMANDS

- **Back** emphasizes data nodes going backwards from a distant abstract node to the point of view.
- **Zoom** Centers and magnifies the screen image appropriately on a node or group of nodes

10

## SUPPORTED APPLICATIONS

- **Mail** sends email to an address that is either explicitly selected, or begins a dialogue to choose one or more addresses based on their proximity to the current point of view. This is an example of a command which uses information from neighbor-values such as type and distance to make decisions. Uncertainty is resolved by the user who selects from a list of candidates proposed by the application.

15

20

- *Simple Tabular Presentation (Spread-sheet format)*
- 5     • **Activate** runs the most appropriate program on a selected data node. Exactly which program is easily determined and changed if desired, since it is a functional node connected to the selected data node, and is thus viewable through normal DataSea techniques.
- 10    • **Input** takes text given by the user, parses, time-stamps and stores it into DataSea. A typical example of this would be *ad hoc* notes, such as 'phone-call from Bob about printer problem', or 'phone-number of Mary Smith is 845-1234'.

15     EXAMPLE APPLICATIONS

- **Mail (Node target)** From the target node, search the neighborhood for AN("name"), and use the DN proximal to it. From that DN, search for a DN connected to AN("address"). Similar for other ANs of use to a mail program.
- 20    • **Notes** Entire note is made into a DN, words become ANs with links to the parent DN.

The special syntax

"word1=word2"

creates AN(word1) linked to DN(word2) .

- **SSheet**(Node target)      A tabular representation of data and column headers of linked ANs in the neighborhood of target\_node is built:

5

Collect all DNs linked to target. These represent one row of tabular presentation. Label these with column headers of the names of their directly linked, distal ANs. Each subsequent row is built from DNs linked to ANs and each other.

10

Set all of the VR position variables to appear in DataSea display as tabular format when in VRmode.

15

- **Phone**(Node target) looks in the neighborhood of target for a DN linked to AN("phone number")
- **Dir**(Node target) is a special case of SSheet, and looks specifically for directory-related information.

20

#### DATA ACQUISITION

Any application can store new data into DataSea, e.g. the applications Notes and Email.

Custom programs can translate legacy formats into DataSea linked nodes, e.g. to load information about

PATENT

a file system, the names of files and directories are stored into a tree representation first, then suffix and name can be used to create ANs, then content can be analyzed, e.g. by putting it through the Notes processor.

5

A RDB (Relational Database) would be loaded by storing the names of databases, tables and columns into ANs, and then values into DN's and keys into links. All these would be linked appropriately: e.g. table name linked to column names linked to all DN's having the values in those columns.

10

Web indices and browser histories can be stored.

System resources can be represented in DataSea.

A dictionary or synonym list can be loaded. The Type and Desc of links between synonyms or nodes with similar meaning are set. E.g. Type="synonym", Desc="from Webster's 10<sup>th</sup> Ed."

15

The user need not know about the data structure, such as database tables and their entity relationships in a relational database, or the directory structure of a file system. Nor does the user need to parameterize and decide how to store data, but may rather simply stuff it into DataSea. DataSea will parse the textual data and create links to representing abstract nodes. Abstract nodes are typically single words representing simple or complex concepts, and are linked to data nodes related to them. These nodes typically are massively linked.

20

25

DataSea is accessible from external programs via its API. More interestingly though, Java code may be stored into a node, fully integrating data and methods. The Java code can then act from within DataSea, for instance modifying the rendering of objects or analyzing data and creating new nodes and links.

The sequence of positioning and rendering flows through the network of nodes from the POV distally. Typically an application will start from one node, specified by name, pointing device or other means, and will search the neighborhood of that node for certain relationships or values and types. For example, invoking "Phone Jim" can find the nearest DN(Jim), then present the nearest DN which is linked to AN("phone number"). Thus commands like "Phone emergency" can work since 'emergency' can be linked to '911' which can have a large default CS which allows it to dominate, and "Phone 123 Main St" can work since the address "123 Main St" can be linked to a phone number through a DN of a person's name.

In addition to the DataSea commands such as show, abs and sim, new applications can be written to extend the base command set of DataSea.

All nodes have the capacity to store a VRObject which contains position and rendering information. It includes a triplet of numbers describing the relative position of a child to its parent, if the rendering mode of DataSea is set to 'VR-mode'.

APPLICATIONS IN DATASEA: HOW THEY DIFFER FROM TYPICAL APPLICATIONS OUTSIDE OF DATASEA

Typically computer applications use or set values at *specific locations* of memory and may or may not check their values by some means or rules or comparisons.

DataSea looks for information by nearness (a fuzzy metric) and/or characteristics of its links and/or characteristics of nodes directly or indirectly linked and/or their values.

Besides looking for DNs which are linked to specific ANs, an application in DataSea can query the distance or conceptual distance from a node to one or more values (of values such as data values, TimeStamps or other parameters). Decisions can be based on complex functions of environment checking.

VISUAL PRESENTATION

The visual tools of DataSea are based on a visual language which is completely different from today's standard GUI's and gives the user easier access to relevant data, and inhibits irrelevant data. DataSea can visually present large amounts of data and the relationships amongst them, emphasizing that which is relevant while keeping the larger context. The user sees exactly the data that is needed as well as related data, a form of look-ahead, albeit at lower resolution.

5 The data presentation changes as the user interacts with DataSea. Data moves smoothly from the background to the foreground, bringing it to the users' attention in response to the user. The gradual shift in visual states helps the user to understand what is happening as the query progresses.

10 The scene begins with a sea of objects representing nodes. Ordering of this sea begins as a result of commands to set a POV or by changing the mode to VRmode on some or all nodes. Typically one sees the sea of data in the background with the POV in the foreground and a TimeLine along an edge such as the bottom. Nodes move and change their appearance with interactions. These interactions can be with the user or with programs inside DataSea or externally.

20 The positions of nodes are changed by iterative calculations of forces on them, thus they move visibly between positions, rather than jumping suddenly. In this way changes in state, and thus appearance, can be followed by the user better than by sudden changes of appearance.

#### VISUALIZATION (IN ACCORDANCE WITH THE INVENTION)

25 Nodes are positioned dependent a set of pressures from sources, each pressure from a source (e.g. POV, parent, neighbors) being a function of that source's preferred position or distance between the child and the source, the child's mag, dist, etc.

The optimum distance to point of view is proportional to  $\text{dist}/f(\text{mag})$ .

A node is stationary once these forces are balanced.

5           Rendering is also dependent on mag, dist, and mode.

Point of view is either a new temporary node set at a specific position on screen, or is an existing node.

10           Visual Presentation

15           The visual tools of DataSea are based on a visual language which is completely different from today's standard GUI's and gives the user easier access to relevant data, and inhibits irrelevant data. DataSea can visually present large amounts of data and the relationships amongst them, emphasizing that which is relevant while keeping the larger context. The user sees exactly the data that is needed as well as related data, a form of look-ahead, 20           albeit at lower resolution.

25           The data presentation changes as the user interacts with DataSea. Data moves smoothly from the background to the foreground, bringing it to the users' attention in response to the user. The gradual shift in visual states helps the user to understand what is happening as the query progresses.

For example, compare the ease of understanding either of these two scenarios: First, watching five animated objects, which represent five words in alphabetical order, reverse their order, representing reverse-alphabetical ordering: Second, watching five words on a line change from ascending alphabetical order to descending. In the first case, reversal is apparent. In the second, the simple operation of reversal is far less apparent: seeing the reversal requires re-analyzing the words and then trying out one or more possible explanations. In DataSea, nodes cluster and move individually and in groups in response to queries. Internal parameters inherent in each node and link change in response to queries. These internal parameters are mapped to visual behavior and appearances, such as size, position, color and shape. These visual cues are used to enhance certain nodes or groups of nodes and their links. The internal parameters are changed by (typically recursive) commands that start at one node and spread through links to others. Commands adjust connection strength and magnitude of nodes based on their programmed algorithms and local node and link information, such as node type and the distance from the point of view. The point of view distance parameters are associated with each node and are functions of the shortest path from that node to the point of view. Recursive commands are self-terminating: typically but not always acting distal to the point of view (where the value of the next nodes distance is greater than or equal to the current distance) and often but not always producing less effect further away from the point of view.

The initial appearance of the GUI is a pseudo-3D view of:

- 5       • a backdrop containing the entire data set: The representations here are relatively stable, and provide an orienting reference for the user;
- a timeline along the bottom of the backdrop and
- 10     • a foreground region in which the user creates Points of Views (*point of views*) and into which data are brought forward from the backdrop. The dimension from back to front essentially represents the degree of customization of data presented to the user.

15       A new query is begun by entering words, similar to a web-search, or manipulating regions of the background with the mouse. One or more data nodes are directly 'hit', increasing their magnitude, and secondary nodes (those distal to a primary) and their links are affected: exactly how depends on the spread mode of the operation. Nearness to the point of view is usually a function of link-distance and magnitude, but other methods are possible, e.g. link-distance alone which display data in a simple hierarchical set of 'levels'. Details of nodes are normally suppressed, but with the 'magnifier mode' turned on, any node under the cursor presents more information. Another mode is 'warp mode', which acts like a large magnifying lens on a region of the

20

25

screen. This is similar to hyperbolic viewing of networks of nodes.

Which nodes are enhanced depends on the command and the spread mode, which is the way in which it  
5 traverses the linked nodes. The simplest spread mode is 'radial': this modifies the node at distance  $n+1$  based on the strongest node directly connected to it of distance  $n$ , in effect being influenced by the node which is on the strongest path back to the point of  
10 view. Another spread-mode is 'sum', which adds up all the contributions of nodes of distance  $n$  to directly connected nodes of distance  $n+1$ . In 'sum' mode, a single data node distal to a large number of nodes will sum all their contributions. This is especially  
15 useful in the Similarity and Abstractions operations.

If a specific node is specified in the query, it is enhanced by, for instance, growing in size and moving towards the point of view from the background  
blur of nodes. If an operation of an abstraction  
20 type is used, the abstract nodes are enhanced. The relative positioning of higher or lower levels of abstraction depends on the specific command. If an operation of a similarity type is used, data nodes predominate by approaching the point of view and by  
25 being enhanced.

Rather than connecting the hits immediately and directly to the point of view, abstract-nodes in common are first drawn near the point of view. These  
30 more abstract nodes are then followed by more detailed ones receding back to the backdrop, positioned to give the sense of their being pulled out of the DataSea. Qualities like time since an

event, or distance to one or more chosen abstract nodes can act as a secondary force, or wind, acting to influence the position of nodes along one of the 3 dimensions of the visualization.

5      LINKS

10      Data in DataSea is heavily linked without restrictions on what can be linked. DataSea solves the 'cobweb' visual problem by establishing a point of view for the users' queries. The problem of following links that are loops is solved by calculating, on the fly, the shortest number of links from the point of view to the nodes. This turns a series of self-referencing loops into a temporary hierarchy, based on the current point of view.

15      The user can browse raw data in DataSea, but meaningful structure comes from the interaction between the point of view and raw data. This is analogous to the quantum-physics effect of forcing a wave function into a specific physical state by  
20      applying an observation to the wave function: interaction with the user that forces data into its useful, visible state.

25      A point of view is one form of an abstract node. Once the user finishes a query, the point of view that has been created can be absorbed into DataSea and used later, a form of 'checkpoint' used in calculations.

## PATENT

Links can occur rather mindlessly, for instance simply by association to part or all of an inputted document, in a way which captures relationships, for instance field definitions from legacy databases, or semantic meaning from, for example, some level of natural language processing.

Postprocessing inside DataSea creates abstract nodes. These represent abstractions of the data inside DataSea, representing concepts or the results of analysis. A 'mature' DataSea will contain a large proportion of these abstract nodes.

Each event which links data within DataSea stores a link ID along with it. Thus any two nodes can be linked together more than once, each link having a different ID to differentiate the context of their being linked. A single link ID can be used between many nodes, as long as that particular subset of nodes has a meaningful context. This context is stored in an abstract node which, linked of course to the subset with that link ID, and contains the reason for the links.

## DATA

Data is user-defined and customizable: whatever the user puts into DataSea, it merely needs to be in a computer representation. Data is held inside so-called 'nodes', which may be linked together. A data-node can be a specific value, text such as a web page or free-form entry, or an object representing

5 something as complex as a virtual-reality view of a  
manufacturing facility. Text in any language is  
broken up into words and stored. All of the  
different forms of data share identical mechanisms of  
storage, linkage, search, presentation and access.  
The database contains highly linked data but differs  
in significant ways from RDBMS's (relational database  
management systems), including the ability to create  
links between any data and the elimination of  
10 structured tables. Rather than using pre-defined  
fields to capture relationships, DataSea uses nodes  
with appropriate links. As new data is introduced  
and linked to the existing nodes, alternate paths are  
created between points. This allows data to be found  
15 which contains no keywords contained in the query,  
relying on associations contained in the new data. A  
simple example would be loading a dictionary into  
DataSea: there are few related concepts that are not  
linked through only even two or three definitions of  
20 either. Thus, a user may enter a query containing no  
keywords of a document and be presented with that  
document, albeit emphasized less than documents that  
contain more direct links to the query terms. AI or  
manual 'digestion' of information and linkage to  
25 abstract concepts is of course possible, as is done  
by those who compile databases for search engines  
today.

The user need not know about the data structure,  
such as database tables and their entity  
30 relationships in a relational database, or the  
directory structure of a file system. Nor does the  
user need to parameterize and decide how to store  
data, but may rather simply stuff it into DataSea.

5 DataSea will parse the textual data and create links to representing abstract nodes. Abstract nodes are typically single words representing simple or complex concepts, and are linked to data nodes related to them. These nodes typically are massively linked.

10 DataSea is accessible from external programs via its API. More interestingly though, Java code may be stored into a node, fully integrating data and methods. The Java code can then act from within DataSea, for instance modifying the rendering of objects or analyzing data and creating new nodes and links.

#### APPLICATIONS

15 A fully integrated application in DataSea uses the DataSea linkage and VR mechanisms to provide the functionality of typical window/menu systems. The program of the application is stored in a DataSea application node.

- 20
- The typical steps taken by DataSea applications include:
  - the neighborhood of the target node is searched by the application for application-specific data requirements
  - New formatting nodes are created (eg. A 'page' representing the template for a letter)
- 25

- Links are made to data nodes and their VR positions are set relative to the formatting nodes.

#### VR MECHANISMS

5           All nodes have the capacity to store a 3-D  
vector called a 'VR-position'. This is a triplet of  
numbers describing the relative position of a child  
to its parent, if the rendering mode of DataSea is  
10       set to 'VR-mode'. Any child having non-zero a VR-  
position variable will position itself relative to  
the calling parent based on the VR-position values.

#### ESSENTIAL INTERNAL ELEMENTS OF DATASEA

15       In a preferred implementation, DataSea is a  
pure-Java application. Once loaded, user-defined  
data-nodes and links are used to visualize  
information from a range of sources in an interactive  
or programmatic way. Data-node sources can be email,  
web sites, databases or whatever is required. Fig. 3  
is a block diagram of an embodiment of the invention.

20       All data is contained in objects called nodes.  
Information describing the data is held in the data-  
node. A complex data-node may be broken into smaller  
ones. A data-node has a set of standard fields  
describing itself and any number of links to other  
25       data-nodes.

The DataSea database is a highly linked structure of nodes. A link contains information describing itself and how it relates the linked data-nodes. It therefore contains *semantic* information, adding a new dimension to interactive or programmed processing of data. That is, DataSea supports not just parametric searches (which find the values at certain storage locations specified by parameters) or content-based analysis (which find particular values and their relations anywhere in the database), but the *meaning* of a collection of nodes. An example of this could be a link with the description "located near" relating a computer with a person's name.

Processing of data occurs not only on values of certain parameters, but on any value, independent of what it is describing. For instance, one may search for all information related to an individual's name without specifying which table and column of the database to search, and in which tables and columns to look for foreign keys.

Applications can run inside DataSea, in fact these applications are themselves held inside a node. Current applications such as automatic report generators and data formatters, know which predefined data fields to place, just where, and how to order the values. This functionality is served by DataSea's mechanism of node and link descriptors, which can act as the column names of RDBMS's. The DataSea link description however also provides semantic information about those relationships.

PATENT

Objects are positioned and rendered strongly dependent on their content and their links. That is, features of the rendering of nodes and the relative positions of nodes depend on content and links.

5 Thus, DataSea is unique because the presentation is strongly dependent on the data itself.

USAGE SCENARIO

Below is a scenario of events with comparisons between two different application approaches: The user routinely stores information and calls it up later when faced with a decision as to repair a new printer or buy an old one. This example shows the simplicity and time saved with DataSea. It compares:

- 1) DataSea, and
- 2) A mix of applications consisting of "Outlook Express", "Excel", and "Internet Explorer" in a Microsoft Office Suit, along with two other applications, "Tracker", a call-tracking application and a database front-end application called DB-Front-End here. The numbers of seconds in parentheses following these two methods are estimates of time needed by the methods, in addition to the event itself.

<i>Event: The user receives email from a friend who mentions her new H.P. printer</i>	
<b>Office Suite:</b> email is stored in Outlook Express.	<b>DataSea:</b> email headers and text are automatically stuffed into DataSea.
<i>Event: The user surfs the web and finds advertisement for HP Printer</i>	
<b>Office Suite:</b> Internet Explorer saves the non-overlapping history of URLs temporarily, and relies on the user to bookmark special URLs, and put them in the tree hierarchy defined by the user.	<b>DataSea:</b> with links to DataSea from the browser, each URL visited is stored into DataSea.
<i>Event: The user gets a phone call and makes a note to himself that repairman Bob Smith says that printer A will cost \$300 to repair, and that it is in Joe Baker's office.</i>	

5

**Office Suite:** The user opens the call tracking program 'Tracker' and fills in the fields prompted by the wizard, including the note text "Repairman Bob Smith called...".

**DataSea:** The text of the note is stuffed into DataSea, and explicitly enters: "(Printer A) (office=Joe Baker)". The information is parsed and time-stamped automatically.

10

To store the location of Printer A in a company-wide database, the user invokes the database editing application DB-Front-End, selects appropriate view (e.g. Machine\_View), searches for 'Printer A', enters 'Joe Baker' for the column 'Location'.

15

*Event: The user now wonders if he should replace printer A with a new one. He remembers seeing a reference in a recent email for an HP printer, and also an HP ad on the web, but can't remember exactly where he filed this information.*

20

**Office Suite:** User opens Outlook Express, tries to recall the name of the email sender, possibly keywords to search and sets the time range to search (enlarged since an event 1 minute outside the range will be excluded). Immediately he sees messages focussed on one keyword. User skims header and text to decide if this is the correct message.

**DataSea:** User starts a point-of-view with the initial associative words "'Joe Baker', Printer, email' and gives his guess of when this all occurred via mouse drag on the time-line. He sees several concept-nodes and some data-nodes. He then judges these by emphasizing/de-emphasizing particular ones, and sees email with appropriate links. He further judges them, adds the word "URL" to the point of view which results in the appropriate URL and data being pulled forward

25

Then, user opens Internet Explorer and browses the names in the History list, trying to recall the context for each as he sees them, or tries to recall the name of the document corresponding to the right URL.

30

35

He then deduces which database stored procedure, table or view to use, opens the DB-Front-End application, enters 'Joe Baker' in the correct search field and sees "Printer A" in the Equipment column. He arranges the four windows from these applications for simultaneous viewing (Outlook, Explorer, Tracker, and DB-Front-End).

Example (Sales Pitch): Laptop starts DataSea, voice interface enabled.

User says 'show Mail' ... mail nodes swell, abstract nodes visible.

5           User says 'show Files' 'Tax 97' brings directories forward and shows files.

User says 'show John Smith' which crates a point of view, linked to abstract node named "John Smith"

10           User says 'BMW' which shows Smith's "BMW 528 1985" by virtue of abstract node "car" linked to "BMW", "Ford".

User says 'show address' bringing "123 Main St." forward.

15           User says 'reset', then 'show address' and sees names and addresses of all entries.

User marks timeline over the past week, and says 'show printer and email and Hewlett Packard' which shows an abstract node "Printer" linked to email message about printers and a web page of HP printers

20           User says 'input "John Smith telephone 848-1234" which creates a node holding the entire message, and parses it into smaller data nodes.

User says 'show John Smith'; one sees his telephone number.

## PATENT

To demonstrate abstract nodes and learning: have processed 50 URLs from 'cat' web search,. See all 50 around the abstract nodes surrounding the point of view named 'show cat'. User deselects URLs not related to technical descriptions, the abstract nodes change, bringing forward URLs with more technical information.

## BENEFITS

- 1) Immediate visualization of user-defined data.
- 2) Quick visual feedback on relevant data.
- 3) Less time required to interpret complex data.
- 4) Higher user productivity because DataSea is an intelligent organizer of data.
- 5) Non-technical user can view data in way they understand, not the way the database may be organized.
- 6) Reduces dependency on programmers.
- 7) Reduced bug-count and time for programmers.
- 8) Simpler usage model through single tool to manage and visualize information.
- 9) Time is saved in storing and retrieving information.

PATENT

- 10) Databases can be joined automatically without custom code.
- 11) Points of view can themselves be stored into DataSea, storing interactions with the computer.
- 5 12) Queries and processing results can be stored into DataSea, and used as any other data.
- 10 13) DataSea learns by example: The user may search for data based on relationships to known data or high-level concepts. Judging can be applied to specific data, such as documents, or to concepts, resulting in 'learning by example': the mechanisms of positive and negative feedback to the system are the same.
- 15 14) New data is automatically integrated: New data can be entered and automatically integrated, allowing non-programmers to store data without adapting to the database.
- 20 15) Interoperability issues are moot: Programs can be integrated into DataSea as well as simple data. Since all links between nodes use the same mechanisms, any program has access to any data.
- 25 16) All data can be viewed while maintaining orientation and context: The user can always quickly orient themselves, sparing confusion because data is viewed from the point of view that the user has designated. Context is maintained by position and rendering cues, which indicate the sources of the data and their

5 immediate relationships. The background with its clusters of data-nodes is relatively stable and familiar, and as data is pulled out from it towards the foreground point of view, the data's position is influenced more and more strongly by the criteria of the point of view and nodes connected strongly to it. The user 'judges' nodes: emphasizing a node will enlarge it and bring it to a more noticeable position.

10 Fig. 4 is a screen-shot which shows the result of entering the simple command 'show rocky' ("rocky" representing the name of a user, who has previously entered data pertaining to himself into the system).

15 Fig. 5 is a screen-shot which shows the result of entering 'abs' (when the Fig. 4 display has been generated), bringing forward the abstract nodes which are distal to the data node 'Rocky'. Note the abstract node 'Directory', which, because it groups the abstract nodes 'phone' and 'address' and is thus at a higher level of abstraction, is positioned  
20 closer to the point of view than 'phone' and 'address' abstract nodes.

25 Fig. 6 is a screen-shot which shows the result of entering 'back phone' (when the Fig. 4 display has been generated), bringing the data node between the abstract node 'phone' and the target node 'Rocky' forward.

Fig. 7 is a screen-shot which shows the result of entering 'SS' (when the Fig. 4 display has been generated), which gives a simple 2-column spreadsheet based on the current target data node 'Rocky'.

5            Fig. 8 is a screen-shot which shows the result of entering 'show egg-tempera'. It shows the primary abstract nodes. But what if we want to see some examples of data nodes which are similar to 'Egg-Tempera'? If so, one could enter the command 'show  
10 egg-tempera', 'similar', resulting in the screen-shot of Fig. 9. It is apparent by comparing Fig. 9 with Fig. 8 that the node "Frescoes" and nodes "Glazing\_techniques", "acrylics" and "oils" are brought forward in Fig. 9, near the target data node  
15 "Egg-Tempera".

Aspects of the invention include the following:

1.    Methods of automatically creating a highly connected network of nodes containing data from computer-readable sources. Information  
20    contained in the structure of legacy databases is captured. All data can be integrated. The nodes are identical in structure, as are their links, differing only in their content.
2.    Methods to interactively explore, access and  
25    visualize information in a highly connected network of nodes. These involve setting a point

of view, linking some number of nodes directly  
to it and calculating individual link distances  
from all data nodes back to the point of view.  
This creates a hierarchical network amenable to  
5 visualization even though there may be cyclic  
loops in the links. This hierarchy may change  
whenever a link is added or deleted. Other  
internal parameters such as the connection  
10 strength of each link and the magnitude of each  
node are used in the visualization to calculate  
position and size of each node.

These methods:

- A. emphasize relevant data throughout the  
query process;
- 15 B. are tolerant to imprecision and errors in  
queries. This ability improves as the data  
set grows;
- C. allow access directly, or indirectly;  
20 retrieving relevant data containing none of  
the key-words used in the query;
- D. allow finding data similar to known data,  
without specifying its characteristics;
- E. give smooth changes in visual state rather  
25 than step-wise changes, and provide  
information to the user in the manner that

the nodes move (speed and direction) and appear (size, color);

F. show available categories that a particular datum is a member of;

5 G. integrate virtual reality renderings when appropriate;

10 3. Method of breaking display space into an array of cells, having dimension one more than the dimension of the space displayed on the screen, the extra dimension being size. These are linked to nodes and used by the user interface to rapidly access individual or groups of nodes.

Additionally these methods:

A. are accessible to the naïve user;

15 B. allow emulation of applications such as relational databases and spreadsheets;

C. use a simple command and query syntax which is amenable to a voice interface;

20 D. use time efficiently: user spends time using commands that act directly on data, rather than time spent navigating a pull-down menu interface.

- E. focus time spent on becoming expert on the data set, rather than the user interface.

Variations on the preferred embodiment include:

5 Variation 1: Voice integration. Front end routines take either keyboard input or voice input, submitting word strings from either to handler functions. Voice word 'go' acts as keyboard 'Enter'.

Variation 2: Client server, a wireless or wired client, display mode set to abbreviate early.

10 Self Diagnostics and Use as a Debugger:

15 DataSea can be used to visualize the DataSea program itself. Besides visualizing nodes which represent data for the user, as described elsewhere in this document, in so-called 'dataset nodes' the nodes that are visualized in DataSea can represent internal programming objects, methods or elements of DataSea itself (providing a sort of built-in debugger).

20 Code can be inserted into the program which will visualize each method's invocation and its modifications of user data.

DataSea separates the two tasks of modifying the values of node variables and rendering of those nodes. Thus DataSea can redraw the entire scene not only after traversing the linked nodes and  
5 re-calculating their internal parameters, but the entire scene can be re-drawn at any time during these calculations, even once every time a dataset-node variable such as 'mag' is changed.

Thus, a self-node can indicate to the user its  
10 own activity, by redrawing the entire scene normally and then highlighting itself, or drawing lines to a dataset-node or its elements that it is operating on.

For instance, if a user commands DataSea to increase the variable 'mag' of a node, the method  
15 which does that (e.g., 'spread()') can draw a line from the self-node representing 'spread()' to the dataset node it is modifying. A simple implementation could be as follows:

If the method spread() recursively calls the  
20 method spread\_recursive(), insert a conditional call to touch() after spread\_recursive:

```
spread(Node node) {  
  
    // for all children of node  
  
    // spread_recursive(child);  
  
25    // touch(node, child) }
```

where 'touch(Node caller, Node target)' will visualize the accessing and setting of variables in the target, where 'caller' is the spread() self-node and 'target' is the dataset node being operated on.

5           The method touch(Node caller, Node target) could be implemented as follows:

```
touch(Node caller, Node target) { // Show a line
between caller and target nodes

clear_screen(); // clear the screen

10       render_all(); // render all the nodes normally

draw_line_between_nodes( caller, target ); // draw a
line

sleep(500); // pause so user can follow what is

          happening }.
```

15           Aspects of the invention include:

          a method and apparatus for creating nodes containing data, linking the nodes into a network, setting parameters of the nodes (node variables, and maintaining information specific to each node, e.g. mag, CS, direction of the link (polarization). Each node preferably has a name associated with which it can be searched from a master list;

20

5 a method and apparatus using "context nodes" to modulate link connection strength (CS) and establish context for groups of nodes. For example, a method for associating a set of links and establishing a context node which can modulate the CS of those links thereby sensitizing or desensitizing them to further operations. The context node can also magnify the nodes linked by each link it modulates;

10 a method and apparatus for loading data from free-form notes. For example, a method of taking text input (text from user or application, or text resulting from voice translation) and establishing a set of linked nodes therefrom by:

15 creating a new node for the full text called the full-text-node;

discarding selected words (e.g. articles)

linking the full-text-node to individual nodes representing each remaining word in the full text, creating new nodes as needed.

20 For another example, a method of converting tabular data, i.e., text organized into rows and columns, with column headings (or RDMBS data, with additional links for the keys of the RDBMS), into a set of linked nodes, in which:

25 each column heading is represented by an AN, the column-heading-AN

each cell of data is represented by a DN

links are established between each column-heading-AN (representing a particular column) and those nodes corresponding to the cells in that column

5 links are established between those nodes corresponding to each cell in a row from the table.

For another example, a method of converting files from a computer file system or a set of files linked by HTML references into a set of linked nodes, in which:

10 DNs are established representing each directory or file;

15 links from each node are established to terms found in the file content, e.g., as is done in the parsing of notes. The procedure can filter the content looking for only certain tag values such as meta-tags or heading values (e.g. <H1> Title Here </H1> has "Title Here" as heading-level-1 in HTML)

20 Another set of links can be made to ANs representing the suffix of files, or such ANs can be used as ContextNodes for all links to those files.

For HTML files, links are to be established between nodes representing HTML files and other nodes representing HTML files that are referenced by the first HTML file.

25 For another example, a method of converting files from a computer file system, in which links are

established between DNSs representing file directory with DNSs representing files or sub-directories in that directory;

5 a method and apparatus for defining a POV (either a particular node or a new node linked to a particular node);

10 a method and apparatus for defining distance (as a function of the number of links between nodes and the node type) and hierarchy from the POV and determining distal and proximal directions, in which: once a POV is set and distances calculated from it, a hierarchical 'tree' is defined from what was an arbitrarily complex cross-linked network of nodes. Thus, if any node 'x' has had its distance set by this routine, one is guaranteed to find a path from that node 'x' back to the POV by traveling on a path between nodes of ever-decreasing distance values;

15 a method and apparatus for retrieving data which is linked into a network of nodes interacting with the user to better present the desired data;.

20 a method for emphasizing nodes and paths by tracing backwards from a target node to a POV by following all links to nodes whereby the next node has magnitude less than that of the prior node. Emphasizing those nodes on the path(s) shows nodes 'between' the target and POV. By traveling backwards from the target node to the POV, there may be more than one node having a distance less than the target. This is fine, and if all paths backwards (with the requirement that they are consistently proximal) are

emphasized it is fine. For example, with Bob being the POV, and traveling backwards from the node representing January 1999, all nodes such as notes and events related to Bob will be emphasized;

5           a method for assigning position to each node which is dependent on the node's parameter values, including distance, CS and magnitude. Rather than setting the node at the calculated position immediately, it moves there gradually thereby showing  
10 the transition between states. One way to do this is to calculate forces on a node which are related to the difference between the node's current position' and an ideal calculated position.

15           The ideal position depends on the positioning mode in effect:

          a Relations Mode: Most suitable for narrow queries where we wish to see all the links between nodes in the target data set. Nodes fan out from their parent; the angle dependent on the number of children  
20 their parent has, their distance dependent either on (mag) or (1/mag); or

          a Levels Mode: Most suitable for broad queries where there are too many links between nodes in the target data set. Starting in the center of the  
25 screen, fanning out to the left dependent on their distance from the POV, ANs are rendered. Starting in the center and belonging to the right half of the screen are the DNS whose position moves further to the right the lower their mag;

a method and apparatus for visualizing data, by appearance on a screen. For example, a method of assigning visual emphasis (color, size) to each node dependent on the nodes distance, CS and magnitude.

5           Examples of operations performed on nodes of the inventive set of linked nodes (or on a sea of displayed representations of such nodes) include:

10           'ABS' (for characterizing and understanding the environment of nodes and their ANs): from a target node, traveling distally and upstream, find the first AN and emphasize it. This 'abstracts' the target node in terms of linked ANs. To abstract it at a higher level, go from those ANs to directly linked ANs which are both distal and upstream. This can continue to  
15           arbitrary level until we run out of nodes (realistically not very far, a handful of levels);

20           'XABS' (for emphasizing ANs from a group of nodes, those ANs not having been recently visited by query operations: emphasizing distally from these ANs will result in a relatively large number of DNs being modified. The user may find ANs which are obviously related or not related to their interest, and thereby significantly change the presented data set. Since  
25           these ANs haven't been used recently, we in effect triangulate the target data set from more vantage points. Determining categories which, when evaluated by the user as good or bad, have a large effect on narrowing the presented data set, that is helping the user find the target data set;

5       'SIM': a method of emphasizing (magnifying)  
nodes based on their similarity to a chosen node  
without specifying values of any node (using the  
'sim' command which emphasizes DNs linked to any or  
all of the ANs which are linked to the chosen  
node(s));

10       "POTMAG": a method of modifying the variable  
Potentiation of a node, and using that value to  
influence the degree of change to the variable 'mag'  
from a subsequent operation. Thus, one operation on  
the first set of nodes may call  
Node1.setPotentiationValue() and a subsequent  
operation on the second set of nodes may set the  
value of Node1.mag based on  
15       Node1.getPotentiationValue(). This 'primes' a set of  
nodes, and can operate approximately as a soft, or  
non-binary AND operation.

20       Another aspect of the invention is structuring  
of a set of linked nodes (a "network") including  
"application nodes" (sometimes referred to as  
applications). Applications are nodes containing  
code which get the information they operate on from  
traversing the network. E.g. an email node-  
application is linked to, or given a reference to,  
25       the node "Bob Smith", and upon being invoked (by the  
action function inherent in each node or otherwise)  
searches the neighborhood of the "Bob Smith" node for  
a DN linked to an AN representing email address. If  
more than one is found, the user is presented with  
30       the selection to choose from. Thus any node-  
application can be 'applied' to any node.

One aspect of the invention is a method of accessing data, wherein the data is structured as a set of linked nodes, and each of the nodes includes at least one link to another one of the nodes. The method includes the steps of:

preliminary to displaying representations of the nodes on a screen in a screen space having N dimensions, where N is an integer, dividing a display space having dimension N + 1 into an array of cells, wherein the dimension of the display space includes a size dimension;

linking each of the nodes to at least one of the cells; and

implementing a user interface which displays representations of at least some of the nodes on the screen having sizes determined by the cells to which said at least some of the nodes are linked, wherein the user interface rapidly accesses individual ones or groups of the nodes in response to selection of at least one of said representations.

What follows is a source code listing (in the Java programming language) of a computer program for programming a computer to implement an embodiment of the invention. In the listing (which consists of parts labeled "TL.java," "Timer.java," "ColorObj.java," "Link.java," "Mode.java," "Node.java," "Force.java," "GetURLInfo.java," "Input.java," "Populate.java," "GUI.java," "DataSea.java," "LinkObj.java," "VRObj.java," and "nsr.java"), the object "gui" of class "GUI" is the

PATENT

top-level object, and instantiates the object  
"datasea" of class "DataSea" (and other objects).

05/22/00  
22:48:22

## Confidential and Proprietary Property of Rocky Nevin

TL.java

1

```
// This is TL.java by Rocky Nevin

import java.lang.*;
import java.awt.*;
import java.util.*;
import java.sql.*; // for the class Timestamp
import java.io.*;

/*
 * class TL (Timeline)
 * init() initializes String month_names[]
 * create_date_node() takes date string, finds or creates&links smallest appropriate time-node
 */

class TL extends Object {

    static String month_names[];
    static String day_names[];
    static String hour_names[];
    static String year_names[];
    static GUI gui;

    public void init (GUI passed_gui) {

        gui = passed_gui;

        month_names = new String[12];
        day_names = new String[31];
        hour_names = new String[24];
        year_names = new String[10];

        month_names[0]="Jan"; month_names[1]="Feb"; month_names[2]="Mar";
        month_names[3]="Apr"; month_names[4]="May"; month_names[5]="Jun";
        month_names[6]="Jul"; month_names[7]="Aug"; month_names[8]="Sep";
        month_names[9]="Oct"; month_names[10]="Nov"; month_names[11]="Dec";

        System.out.println("TL init() done.");
        return;
    }

    /*
     * month_from_double
     *
     * return string of month based on double between 0 and 1
     */
    public String month_from_double (double val) {
        if (((0>val) || (1.0 < val))) {

            System.out.println("month from double() Error: month from double("+val+")");
            return("["month_from_double() Error: month from double("+val+")"]);
        }

        val *= 12;
        val *= 0.9999; // 0 OK, 12 not OK
        int m = (int)val;
        return(month_names[m]);
    }

    /*
     * create_TS
     *
     * return Node based on String s linked to caller
     */
    public Node create_TS (Node caller, String CNodeName, String s) {
        Node ts_node = create_date_node(s);
        Node CNode = gui.database.find_node_named(CNodeName, "CN");
        if (CNode == null) {
            //System.err.println("create_TS() Error: null CNode for caller <"+caller.Name
            +">, string = <"+s+">");
            caller.link(ts_node);
            if (caller == ts_node)
                System.out.println("create_date_node ERROR(1): caller == ts_node <
                "+ts_node.Name+">");
        }
        else {
            caller.link(ts_node, CNode, "polarized");
            if (caller == ts_node)
                System.out.println("create_date_node ERROR(2): caller == ts_node <
                "+ts_node.Name+">");
        }
        return(ts_node);
    } // end create_TS

    /**
     * find_day_in_string return a string which is a date, like 1,2...31
     */
    public String find_day_in_string (String s) {
        int i, size;
        String numbers = " 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 ";
        numbers = numbers+" 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 ";
        String[] words = gui.input_string.to_array(s);
    }
}
```

05/22/00  
22:48:22

## Confidential and Proprietary Property of Rocky Nevin

TL.java

2

```
size = words.length;
for (i=0; i<size; i++) {
    if (0<=numbers.indexOf(words[i]))
        return(words[i]);
}

return((String)null);
} // end find_day_in_string

/*
 * create date node
 *
 * search for year, month, day, time, create node for particular date
 * using finest resolution appropriate, create nodes for time foundation
 * -create day/month/year offsets
 * -create lowest/highest resolution nodes, link them
 */

public Node create_date_node(String s) {
    double x=0, year_offset=0, month_offset=0, day_offset=0, day_value=0;
    boolean year_found=false, month_found=false, day_found=false;
    Node year_node=null, month_node=null, day_node=null;
    Node finest_grain_node=null; // The finest-grained TL node to which we'll link DN to
    Node coarsest_grain_node=null; // The largest-grained TL node to which we'll link Timeline to
    Node last_node=null;
    String year_str="";
    String month_str="";
    String day_str="";
    String rel_str="";
    double TimelineSizeY = gui.datasea.pop.Timeline.size.Y;
    double TimelineSizeX = gui.datasea.pop.Timeline.size.X;
    double TimelineX = gui.datasea.pop.Timeline.X;
    double TimelineY = gui.datasea.pop.Timeline.Y;
    double base_year_offset = 1.0/2;
    double base_month_offset = base_year_offset/12;
    double base_day_offset = base_month_offset/31;
    double default_size_X = 2;
    int delta_x = 2; // used as offset between year, month, day

    //-----
    // LOOK FOR YEAR
    if (0 <= s.indexOf("1999")) {
        year_str = "1999";
        year_offset = 0;
        year_offset = base_year_offset * 0 * TimelineSizeY;
        year_found = true;
    }

    if (0 <= s.indexOf("2000")) {
        year_str = "2000";
        year_offset = base_year_offset * 1 * TimelineSizeY;
        year_found = true;
    }

    //-----
    // LOOK FOR MONTH
    if (0 <= s.indexOf("Jan")) {
        month_offset = base_month_offset*0 * TimelineSizeY;
        month_str = "Jan";
        month_found = true;
    }

    if (0 <= s.indexOf("Feb")) {
        month_offset = base_month_offset*1 * TimelineSizeY;
        month_str = "Feb";
        month_found = true;
    }

    if (0 <= s.indexOf("Mar")) {
        month_offset = base_month_offset*2 * TimelineSizeY;
        month_str = "Mar";
        month_found = true;
    }

    if (0 <= s.indexOf("Apr")) {
        month_offset = base_month_offset*3 * TimelineSizeY;
        month_str = "Apr";
        month_found = true;
    }

    if (0 <= s.indexOf("May")) {
        month_offset = base_month_offset*4 * TimelineSizeY;
        month_str = "May";
        month_found = true;
    }

    if (0 <= s.indexOf("Jun")) {
        month_offset = base_month_offset*5 * TimelineSizeY;
        month_str = "Jun";
        month_found = true;
    }

    if (0 <= s.indexOf("Jul")) {
        month_offset = base_month_offset*6 * TimelineSizeY;
        month_str = "Jul";
        month_found = true;
    }

    if (0 <= s.indexOf("Aug")) {
        month_offset = base_month_offset*7 * TimelineSizeY;
        month_str = "Aug";
        month_found = true;
    }
}
```

05/22/00  
22:48:22

# Confidential and Proprietary Property of Rocky Nevlin

TL.java

3

```

    )
    if (0 <= s.indexOf("Sep")) {
        month_offset = base_month_offset*8 * TimelineSizeY;
        month_str = "Sep";
        month_found = true;
    }
    if (0 <= s.indexOf("Oct")) {
        month_offset = base_month_offset*9 * TimelineSizeY;
        month_str = "Oct";
        month_found = true;
    }
    if (0 <= s.indexOf("Nov")) {
        month_offset = base_month_offset*10 * TimelineSizeY;
        month_str = "Nov";
        month_found = true;
    }
    if (0 <= s.indexOf("Dec")) {
        month_offset = base_month_offset*11 * TimelineSizeY;
        month_str = "Dec";
        month_found = true;
    }

//
// LOOK FOR DAY
if ((day_str != null) && (day_str.indexOf(" ") != -1)) {
    day_value = (double)(GUI.java.lang.Double.parseDouble(day_str));
    day_offset = base_day_offset * (day_value-1) * TimelineSizeY;
    day_found = true;
}

//
year_offset += TimelineY;
month_offset += year_offset;
day_offset += month_offset;

//
if (day_found) {
    ret_str = ret_str + day_str + " ";
    if (day_value == 2 && month_str == "Mar")
        GUI.P(0, "create_date_node", "----> day 2 for <"+day_str+" "+month_str+" "+year_offset);
    day_node = gui_datasea.pop_create_node(day_str + " "+month_str + " "+year_offset, "Event");
    //day_node = gui_datasea.pop_create_node_forced(day_str + " "+month_str + " "+year_offset, "Event");
    day_node.Y = day_offset;
    last_node = day_node;
    finest_grain_node = day_node;
    coarsest_grain_node = day_node;
}

if (month_found) {
    ret_str = ret_str + month_str + " ";
    month_node = gui_datasea.pop_create_node(month_str + " "+year_offset, "Event");
    if (last_node != null) {
        month_node.link(last_node);
        if (last_node == month_node)
            System.out.println("create_date_node ERROR: last_node == month_node <"+month_node.Name+">");
    }
    month_node.Y = month_offset;
    last_node = month_node;
    if (finest_grain_node == null)
        finest_grain_node = month_node;
    coarsest_grain_node = month_node;
}

if (year_found) {
    ret_str = ret_str + year_str;
    year_node = gui_datasea.pop_create_node(year_str, "Event");
    if (last_node != null) {
        year_node.link(last_node);
        if (last_node == year_node)
            System.out.println("create_date_node ERROR: last_node == year_node <"+year_node.Name+">");
    }
    year_node.Y = year_offset;
    last_node = year_node;
    if (finest_grain_node == null)
        finest_grain_node = year_node;
    coarsest_grain_node = year_node;
}

if (year_node != null) {
    year_node.X = TimelineX+delta_x;
    year_node.size.X = default_size_X;
    year_node.size.Y = 0.95 * TimelineSizeY * base_year_offset;
}

if (month_node != null) {
    month_node.X = TimelineX+2*delta_x;
    month_node.size.X = default_size_X;
    month_node.size.Y = 0.95 * TimelineSizeY * base_month_offset;
}

if (day_node != null) {
    day_node.X = TimelineX+3*delta_x; // + (31-day_value);
    day_node.size.X = default_size_X;
    day_node.size.Y = 0.95 * TimelineSizeY * base_day_offset;
}
```

05/22/00  
22:48:22

Confidential and Proprietary Property of Rocky Nevin

TL.java

4

```
// day_node.Y += (Math.random()-0.5);  
}  
  
Node event_node = gui.datasea.pop.create_node(ret_str, "Event");  
event_node.X = finest_grain_node.X+2;  
event_node.Y = finest_grain_node.Y;  
event_node.size_X = default_size_X;  
event_node.size_Y = 0.95 * TimelineSizeY * base_day_offset;  
  
// link to finest grain time node ...  
  
if (event_node == finest_grain_node)  
    ; //System.out.println("create_date_node ERROR: event_node == finest_grain_node <" + fin  
    est_grain_node.Name + ">");  
else  
    event_node.link(finest_grain_node);  
  
// link to Timeline node ...  
gui.datasea.pop.Timeline.link(coarsest_grain_node);  
  
return(event_node);  
} // end create_date_node  
  
} // end class TL (Timeline)
```

05/22/00  
22:47:14

// This is Timer.java by Rocky Nevin

} // end class Timer

```
/*
 * class Timer
 */
class Timer extends Object {
    long TS1=0, TS2=0;
    String TS1_name, TS2_name;
```

```
// CONSTRUCTOR
Timer() {
}
```

```
/**
 ** start_timer
 **
 */
```

```
    public void start_timer (String timer_name) {
        int i;
```

```
        TS1 = java.lang.System.currentTimeMillis();
        TS1_name = timer_name;
    } // end start_timer
```

```
/**
 ** end_timer
 **
 */
```

```
    public void end_timer (String timer_name) {
        int i;
```

```
        if (timer_name.equals(TS1_name)) {
            TS2 = java.lang.System.currentTimeMillis();
            TS2_name = timer_name;
```

```
            if (GUI.Debug == -1)
                GUI.P((-1), "end_timer", "Time for "+TS1_name+" is "+(TS2-TS1)+" millisec
onds");
        }
```

```
    else
        if (GUI.Debug == -1)
            GUI.WARNING((-1), "end_timer", "Wrong name: TS="+TS1_name+", TS2="+
            TS2_name);
    } // end end_timer
```

05/22/00  
22:48:41

# Confidential and Proprietary Property of Rocky Nevin ColorObj.java

1

```
// This is ColorObj.java by Rocky Nevin
import java.awt.*;

class ColorObj extends Object {
    static Color VeryLightGrey, LightGrey, Grey, DarkGrey, VeryDarkGrey, VeryLightRed, LightRed,
    DarkRed, Crimson, VeryLightGreen, LightGreen, Green, DarkGreen, Blue, VeryLightBlue,
    LightBlue, DarkBlue, DarkYellow, Yellow, LightYellow, VeryLightYellow;

    static int MAX_BACKGROUND_COLORS = 4;
    static int background_color_index=0;
    static Color background_color_array[] = new Color[MAX_BACKGROUND_COLORS];

    // Color TempColor;
    static Color RedArray[] = new Color[16];
    static Color PurpleArray[] = new Color[16];
    static Color BlueArray[] = new Color[16];
    static Color GreenArray[] = new Color[16];
    static Color BlackArray[] = new Color[16];

    public void ColorObj () {
        init();
    }

    public void init () { // ColorObj.init
        int i;
        background_color_array[0] = new Color(0x444444);
        background_color_array[1] = new Color(0x112211);
        background_color_array[2] = new Color(0x111122);
        background_color_array[3] = new Color(0xeeecee);
        VeryLightGrey = new Color(0xeeeeee);
        LightGrey = new Color(0xaaaaaa);
        Grey = new Color(0x777777);
        DarkGrey = new Color(0x555555);
        VeryDarkGrey = new Color(0x10a0a);
        DarkYellow = new Color(0x222200);
        Yellow = new Color(0xaaaa44);
        LightYellow = new Color(0xaaaa44);
        VeryLightYellow = new Color(0xffff88);
        VeryLightRed = new Color(0xffddd);
        LightRed = new Color(0xff4444);
        Red = new Color(0xaa4444);
        DarkRed = new Color(0x663333);
        LightRed = new Color(0xff4444);
        Crimson = new Color(0xee2299);

        /*****
        DarkGreen = new Color(0x336633);
        Green = new Color(0x44aa44);
        LightGreen = new Color(0x77aa77);
        VeryLightGreen = new Color(0xddffdd);
        Blue = new Color(0x4444aa);
        LightBlue = new Color(0x777799);
        VeryLightBlue = new Color(0xddddf);
        DarkBlue = new Color(0x333366);

        *****/

        /*****
        for (i=0; i<16; i++)
            RedArray[i] = new Color((i)*0x001111 + 0xcc0000);
        for (i=0; i<16; i++)
            PurpleArray[i] = new Color((i)*0x110011 + 0xcc0044);
        for (i=0; i<16; i++)
            BlueArray[i] = new Color((i)*0x111100 + 0x0000cc);
        for (i=0; i<16; i++)
            GreenArray[i] = new Color((i)*0x110011 + 0x00cc00);
        for (i=0; i<16; i++)
            BlackArray[i] = new Color((i) * 0x111111);
        *****/

        // IF DEMO MODE, for clarity

        for (i=0; i<16; i++)
            RedArray[i] = new Color((i)*0x110000 + (i/2)*0x001111);
        for (i=0; i<16; i++)
            PurpleArray[i] = new Color((i)*0x110011 + (i/2)*0x001100);
        for (i=0; i<16; i++)
            BlueArray[i] = new Color((i)*0x000011 + (i/2)*0x111100);
        for (i=0; i<16; i++)
            GreenArray[i] = new Color((i)*0x001100 + (i/2)*0x110011);
        for (i=0; i<16; i++)
            BlackArray[i] = new Color((i) * 0x111111);

        // BlackArray has 16 elements, 0th is black, 10th is white, we use first 10, index=(10-mag)

        } // end ColorObj::init

    }

    /**
    ** set_color_from_TS
    */
    public void set_color_from_TS (Graphics graphics, Node node) {
        int index=20;
        double Tdiff, val;
        Color color=Color.yellow;
    }
}
```

05/22/00  
22:48:41

## Confidential and Proprietary Property of Rocky Nevin

ColorObj.java

2

```
if (node==null)
    return;
else
    Tdiff = GUI.current_TS - node.TS;

if (node.isForm) {
    graphics.setColor(Color.white);
    return;
}

if (Tdiff < 2000) {
    val = 0.1*(4000.0-Tdiff)/4000.0 * node.delta_mag;
    index = (int)((val + 1.0) * 10); // range is 0 to +infinity

    switch (index) {
        case 0: color = BlueArray[15]; break;
        case 1: color = BlueArray[14]; break;
        case 2: color = BlueArray[13]; break;
        case 3: color = BlueArray[12]; break;
        case 4: color = BlueArray[11]; break;
        case 5: color = BlueArray[10]; break;
        case 6: color = BlueArray[9]; break;
        case 7: color = BlueArray[8]; break;
        case 8: color = BlueArray[7]; break;
        case 9: color = BlueArray[6]; break;
        case 10: color = RedArray[8]; break;
        case 11: color = RedArray[9]; break;
        case 12: color = RedArray[9]; break;
        case 13: color = RedArray[10]; break;
        case 14: color = RedArray[10]; break;
        case 15: color = RedArray[10]; break;
        case 16: color = RedArray[11]; break;
        case 17: color = RedArray[11]; break;
        case 18: color = RedArray[12]; break;
        case 19: color = RedArray[13]; break;
        case 20: color = RedArray[15]; break;
        default: color = Crimson; break;
    }
    graphics.setColor(color);
}

else // neither Tdiff < 4000 nor val!=10
    set_color_from_mag(graphics, node);

return;
} // end set_color_from_TS

/*
** set_color_for_relations
*/
public void set_color_for_relations (Graphics graphics, Node parent, Node child) {
    int index=20;
    double Tdiff, val;
    Color color=Color.yellow;

    if (child==null)
        color = Crimson;
    else if (parent==null)
        color = Red;
    else {
        if (background_color.index < MAX_BACKGROUND_COLORS-1)
            index = (int)(4+Math.floor(child.mag));
        else
            index = (int)(Math.floor(1.5*(10-child.mag)));

        if (index>=0 && index<16)
            color = BlackArray[index];
        else
            if (index<0)
                color = Red;
            else
                if (index>16)
                    color = Blue;
            else
                *****
    }

    switch (index) {
        case 0: color = BlackArray[3]; break;
        case 1: color = BlackArray[3]; break;
        case 2: color = BlackArray[3]; break;
        case 3: color = BlackArray[2]; break;
        case 4: color = BlackArray[2]; break;
        case 5: color = BlackArray[2]; break;
        case 6: color = BlackArray[1]; break;
        case 7: color = BlackArray[1]; break;
        case 8: color = BlackArray[1]; break;
        case 9: color = BlackArray[0]; break;
        case 10: color = BlackArray[0]; break;
        case 11: color = BlackArray[15]; break;
        case 12: color = BlackArray[15]; break;
        case 13: color = BlackArray[15]; break;
        case 14: color = BlackArray[15]; break;
        case 15: color = BlackArray[15]; break;
    }
```

05/22/00  
22:48:41

# Confidential and Proprietary Property of Rocky Nevin ColorObj.java

3

```
case 16: color = BlackArray[15]; break;
default: color = Crimson; break;
}
}
*****
)
graphics.setColor(color);

return;
} // end set_color_for_relations

/*
** set_color_from_mag
*/

public void set_color_from_mag (Graphics graphics, Node node) {
    double mag=0;

    if (node.isMarked) {
        graphics.setColor((Color.cyan).darker());
    } else
    if (GUI.showBoundaries) {
        switch ((int)node.Tdist) {
            case 1: graphics.setColor(DarkGreen); break;
            case 2: graphics.setColor(LightGreen); break;
            case 3: graphics.setColor(VeryLightGreen); break;
            case 4: graphics.setColor(LightRed); break;
            case 5: graphics.setColor(DarkRed); break;
            default: graphics.setColor(Crimson); break; // shouldn't be here
        }
    } else {
        if (node != null)
            mag = node.mag;
        // FIRST SET OF COLORS IS FOR DARK BACKGROUNDS
        if (background_color_index < MAX_BACKGROUND_COLORS-1) {
            if (node.isAN) {
                if (mag >= 0 && mag <= Node.MAX_MAG) {
                    if (mag <= Node.MED_MAG)
                        graphics.setColor(DarkGreen);
                    else if (mag <= Node.BIG_MAG)
                        graphics.setColor(Green);
                    else if (mag <= Node.MAX_MAG-0.1)
                        graphics.setColor(LightGreen);
                    else if (mag >= Node.MAX_MAG-0.1)
                        graphics.setColor(VeryLightGreen); // maximum brightness
                }
            }
        }
    }
}
```

ess

```
else
    graphics.setColor(Crimson); // shouldn't be here
}
}
else
    if (node.isCN) {
        if (mag >= 0 && mag <= Node.MAX_MAG) {
            if (mag <= Node.MED_MAG)
                graphics.setColor(DarkBlue);
            else if (mag <= Node.BIG_MAG)
                graphics.setColor(Blue);
            else if (mag <= Node.MAX_MAG-0.1)
                graphics.setColor(LightBlue);
            else if (mag >= Node.MAX_MAG-0.1)
                graphics.setColor(VeryLightBlue); // maximum brightness
        }
        else
            graphics.setColor(Crimson); // shouldn't be here
    }
}
else
    if (node.isDN) {
        if (mag >= 0 && mag <= Node.MAX_MAG) {
            if (mag <= Node.MED_MAG)
                graphics.setColor(DarkYellow);
            else if (mag <= Node.BIG_MAG)
                graphics.setColor(Yellow);
            else if (mag <= Node.MAX_MAG-0.5)
                graphics.setColor(LightYellow);
            else if (mag >= Node.MAX_MAG-0.5)
                graphics.setColor(VeryLightYellow); // maximum brightness
        }
        else
            graphics.setColor(Crimson); // shouldn't be here
    }
}
}
else
    if (mag >= 0 && mag <= Node.MAX_MAG) {
        if (mag <= Node.MED_MAG)
            graphics.setColor(DarkGrey);
        else if (mag <= Node.BIG_MAG)
            graphics.setColor(Grey);
        else if (mag <= Node.MAX_MAG-0.1)
            graphics.setColor(LightGrey);
        else if (mag >= Node.MAX_MAG-0.1)
            graphics.setColor(VeryLightGrey); // maximum brightness
    }
}
```

S

```
else
    graphics.setColor(Crimson); // shouldn't be here
}
```

```
} else {
// SECOND SET OF COLORS IS FOR LIGHT BACKGROUND
```

```
    if (node.isEvent) {
        if (mag <= Node.MED_MAG)
            graphics.setColor(VeryLightGrey);
        else if (mag <= Node.BIG_MAG)
            graphics.setColor(LightGrey);
        else if (mag <= Node.MAX_MAG-0.1)
            graphics.setColor(Grey);
        else if (mag >= Node.MAX_MAG-0.1)
            graphics.setColor(DarkGrey); // maximum contrast
        else
            graphics.setColor(Crimson); // shouldn't be here
    }
    if (mag >= 0 && mag <= Node.MAX_MAG)
        graphics.setColor(BlackArray((int)Math.floor(1.5*(10-mag)));
    else
        graphics.setColor(Color.black); // maximum brightness
    }
}
//graphics.setColor(BlackArray((int)Math.floor(mag)));
return;
} // end set_color_from_mag
```

```
/*
** set_link_color_from_mag
**
*****
public void set_link_color_from_mag (Graphics graphics, Node node) {
    double mag=0;
    if (node != null)
        mag = node.mag;
    if (mag > 0 && mag < 10)
        graphics.setColor(BlueArray((int)Math.floor(mag)));
    else
        graphics.setColor(BlueArray((int)9)); // default maximum darkness
return;
} // end set link color from mae
```

```
*****
} // End of class ColorObj
}
```

05/22/00  
23:01:29

# Confidential and Proprietary Property of Rocky Nevin

## Link.java

```
// This is Link.java by Rocky Nevin

import java.lang.*;

/*
 * This is Link.java by Rocky Nevin
 * @version 0.4, 3/12/98
 */

public class Link extends Object {
    static public final double DEFAULT_CS = 1.0;
    static public final double MED_CS = 1.0;
    static public final double MIN_CS = 0.1;
    static public final double MAX_CS = 1.0;
    static public final double DELTA_CS = 0.5;
    String Name="";
    String Type="";
    String Desc_R="";
    String Desc_L="";
    double CS_R=DEFAULT_CS, CS_L=DEFAULT_CS;
    Node Node;
    Node CNode=null;
    Node Nodel, NodeR;
    Link NextLink;
    VRObj VR_L, VR_R;
    boolean should_we_pos_L=true, should_we_pos_R=true;
    boolean isPolarized=false; // Bi-directional Link is polarized

    /**
     ** Link CONSTRUCTORS
     **
     */

    public Link() {
        VR_L = new VRObj();
        VR_R = new VRObj();
    }

    // RECURSES this.Node = new Node();
    }

    /**
     ** set_CS
     **
     */

    public void set_CS (double new_CS) {
        this_CS_L = new_CS;

        this_CS_R = new_CS;

        if (this_CS_R > 5)
            this_CS_R = (5+this_CS_R)/2;
        if (this_CS_L > 5)
            this_CS_L = (5+this_CS_L)/2;
        if (this_CS_R < DELTA_CS)
            this_CS_R = DELTA_CS;
        if (this_CS_L < DELTA_CS)
            this_CS_L = DELTA_CS;

        ) // end set_CS

    /**
     ** more_CS
     **
     */

    public void more_CS () {
        this_CS_R += DELTA_CS;
        this_CS_L += DELTA_CS;
        if (this_CS_R > 5)
            this_CS_R = (5+this_CS_R)/2;
        if (this_CS_L > 5)
            this_CS_L = (5+this_CS_L)/2;

        ) // end more_CS

    /**
     ** less_CS
     **
     */

    public void less_CS () {
        this_CS_R -= DELTA_CS;
        this_CS_L -= DELTA_CS;
        if (this_CS_R < DELTA_CS)
            this_CS_R = DELTA_CS;
        if (this_CS_L < DELTA_CS)
            this_CS_L = DELTA_CS;

        ) // end less_CS

    /**
     ** addCNode
     **
     */
```

05/22/00  
23:01:29

## Confidential and Proprietary Property of Rocky Nevin

### Link.java

2

```
*/
public void addCNode (Node CNode) { // CSS
    int i, size;
    Node child;

    //System.out.println("Link.addCNode: adding "+CNode.Name+" to this Link and vice versa");
    this.CNode = CNode;
    if (!CNode.ContextLinks.contains(this)) {
        CNode.ContextLinks.addElement(this);
    }
    // System.out.println("Link.addCNode: NEW: "+CNode.Name+" modulates "+ CNode.Co
    ntextLinks.size() + " links.");
    CNode.isCN = true;
}

//else
// System.out.println("Link.addCNode: ALREADY ADDED: "+CNode.Name+" modulat
    es "+ CNode.ContextLinks.size() + " links.");
} // end addCNode

/**
 ** setLinks.should_we_pos in single-caller version positioning 6/22/99
 */
public void setLinks.should_we_pos (Node node, boolean value) {
    int which=0;

    if (node == null) {
        System.out.println("getNodeNodesVRparams(): ERROR, passed node is null");
        return;
    }

    // determine if we should get Left or Right
    if (node == this.NodeL) // use Left params
        should_we_pos_L = value;
    else
        if (node == this.NodeR) // use Right params
            should_we_pos_R = value;

    return;
} // end setLinks.should_we_pos

/**
 ** should_we_pos in single-caller version positioning 6/22/99
 */
}

*/
public boolean should_we_pos (Node node) {
    int which=0;

    if (node == null) {
        System.out.println("getNodeNodesVRparams(): ERROR, passed node is null");
        return;
    }

    // determine if we should get Left or Right
    if (node == this.NodeL) // get Left params
        which = 1;
    else
        if (node == this.NodeR) // get Right params
            which = 2;

    // now make assignments, make Left negative of Right, since offsets are simple
    switch (which) {
        case 1: // Set link's Left X to node's X, link's Right X to -node's X
            node.X = this.VR.L.X;
            node.Y = this.VR.L.Y;
            break;
    }
}
```

## Link.java

```
case 2: // Set link's Right X to node's X, link's Left X to -nodes's X
node.X = this.VR_R.X;
node.Y = this.VR_R.Y;
break;

default:
    System.out.println("getNodeVRparams(): ERROR, which = "+which);
    break;
}

return;
} // end setNodesVRparams

/**
 ** setlinksVRparams set VR params in link from passed node
 */
public void setlinksVRparams (Node node) {
    int which=0;

    if (node == null) {
        System.out.println("setlinksVRparams(): ERROR, passed node is null");
        return;
    }

    // determine if we should set Left or Right
    if (node == this.NodeL) // set Left params
        which = 1;
    else
        if (node == this.NodeR) // set Right params
            which = 2;

    // now make assignments, make Left negative of Right, since offsets are simple
    switch (which) {
        case 1: // Set link's Left X to node's X, link's Right X to -nodes's X
            this.VR_L.X = node.X;
            this.VR_L.Y = node.Y;
            this.VR_R.X = -this.VR_L.X;
            this.VR_R.Y = -this.VR_L.Y;
            break;
        case 2: // Set link's Right X to node's X, link's Left X to -nodes's X
            this.VR_R.X = node.X;
            this.VR_R.Y = node.Y;
            this.VR_L.X = -this.VR_R.X;
            this.VR_L.Y = -this.VR_R.Y;
            break;
    }

    default:
        System.out.println("setlinksVRparams(): ERROR, which = "+which);
        break;
    }

    return;
} // end setlinksVRparams

/**
 ** dumpLink
 **
 */
public void dumpLink () {
    String nodeName="null", nodeRname="null";

    if (NodeL != null)
        nodeName = NodeL.Name;
    if (NodeR != null)
        nodeRname = NodeR.Name;

    System.out.println("Link.dumpLink: "+nodeName+" ----> "+nodeRname+" X="+this.VR_L.X);
    System.out.println("Link.dumpLink: "+nodeRname+" ----> "+nodeName+" X="+this.VR_L.X);
} // end dumpLink

public class VRObj extends Object {
    // some elements for data, stubbed now
    double X, Y, Z; // relative positions in VR space, typically
    // positions offsets from another node
    double sizeX, sizeY, sizeZ; // sizes
    VRShape Shape; // data and methods to render semi-realistically, for VR

    public VRObj() {
    }

    // end internal class VRObj
}

// End of class Link
```

// This is Mode.java by Rocky Nevin

```
class Mode extends Object {
    String Name="AND";
    String spread_mode="distal";
    String lines_mode="all";
    String position_mode="Rel";
    String render_mode="VR";
    String line_mode="radial";
    boolean do_potential = false;
```

```
    public Mode() {
        super();
    } // end Mode creator
```

```
    /**
     ** set_mode_buttons
     */
    void set_mode_buttons() {
```

```
        //GUI.button_spread.setLabel("Spread "+spread_mode);
        GUI.button_render.setLabel("Render "+render_mode);
        GUI.button_lines.setLabel("Lines "+lines_mode);
        GUI.button_position.setLabel("Pos'n "+position_mode);
        GUI.button_potential.setLabel("Pot "+do_potential);
        if (do_potential)
            GUI.button_potential.setBackground(ColorObj.LightRed);
        else
```

```
            GUI.button_potential.setBackground(ColorObj.LightGrey);
        GUI.button_drawText.setLabel("Text "+GUI.drawText);
        if (GUI.drawText)
```

```
            GUI.button_drawText.setBackground(ColorObj.LightRed);
        else
```

```
            GUI.button_drawText.setBackground(ColorObj.LightGrey);
    } // end set_mode_buttons
```

```
    /**
     ** toggle_draw_text
     **
     */
```

```
    public void toggle_draw_text () {
        GUI.drawText = !GUI.drawText;
        GUI.P1, "toggle_draw_text", " GUI.drawText = "+GUI.drawText);
        set_mode_buttons();
    }
```

```
    /**
     * method set_spread_mode
     */
    void set_spread_mode (String str) {
        spread_mode = str;
        GUI.P1, "Mode.set_spread_mode", "Mode "+str);
        set_mode_buttons();
    } // end set_spread_mode
```

```
    /**
     * method set_render_mode
     */
    void set_render_mode (String str) {
        render_mode = str;
        GUI.P1, "Mode.set_render_mode", "Mode "+str);
        set_mode_buttons();
    } // end set_render_mode
```

```
    /**
     * method set_position_mode
     */
    void set_position_mode (String str) {
        position_mode = str;
        GUI.P1, "ModMode.set_position_mode", "Mode "+str);
        set_mode_buttons();
    } // end set_position_mode
```

```
    /**
     * method toggle_lines_mode
     */
```

```
    void toggle_lines_mode () { // radial, distal, proximal
        GUI.P1, "word selector", "lines_mode = "+lines_mode);
        if (lines_mode.equals("radial")) {
            lines_mode = "distal";
        }
        else if (lines_mode.equals("all"))
            lines_mode = "none";
        else if (lines_mode.equals("none"))
            lines_mode = "local";
        else if (lines_mode.equals("local"))
            lines_mode = "all";
        set_mode_buttons();
    } // end toggle_lines_mode
```

## Mode.java

```
/**
 * method toggle_spread_mode
 */
void toggle_spread_mode () { // radial, distal, proximal
    GUI.P1."word_selector".spread_mode = "+spread_mode";
    if (spread_mode.equals("radial")) {
        spread_mode = "distal";
    }
    else if (spread_mode.equals("distal"))
        spread_mode = "proximal";
    else if (spread_mode.equals("proximal"))
        spread_mode = "radial";
    set_mode_buttons();
} // end toggle_spread_mode

/**
 * method toggle_position_mode
 */
void toggle_position_mode () { // relations, levels
    if (position_mode.equals("Rel")) {
        position_mode = "Grid";
        Force.calcPOVPressure=true;
        Force.calcParentPressure=false;
        System.out.println("toggle_position_mode: setting position_mode to "+position_mode);
    }
    else if (position_mode.equals("Grid")) {
        position_mode = "Lvs";
        Force.calcPOVPressure=true;
        Force.calcParentPressure=false;
        System.out.println("toggle_position_mode: setting position_mode to "+position_mode);
    }
    else if (position_mode.equals("Lvs")) {
        position_mode = "Rel";
        Force.calcPOVPressure=false;
        Force.calcParentPressure=true;
        System.out.println("toggle_position_mode: setting position_mode to "+position_mode);
    }
}

GUI.button_presParent.setLabel("Parent="+Force.calcParentPressure);
if (Force.calcParentPressure)
    GUI.button_presParent.setBackground(ColorObj.LightRed);
else
    GUI.button_presParent.setBackground(ColorObj.LightGrey);
GUI.button_presPOV.setLabel("POV="+Force.calcPOVPressure);
if (Force.calcPOVPressure)
    GUI.button_presPOV.setBackground(ColorObj.LightRed);
else
```

---

```
GUI.button_presPOV.setBackground(ColorObj.LightGrey);
set_mode_buttons();
} // end toggle_position_mode

/**
 * method toggle_spread_mode
 */
void toggle_line_mode () { //
    if (line_mode.equals("radial"))
        line_mode = "distal";
    else if (line_mode.equals("distal"))
        line_mode = "proximal";
    else if (line_mode.equals("proximal"))
        line_mode = "radial";
    set_mode_buttons();
} // end toggle_line_mode

/**
 * method toggle_render_mode
 */
void toggle_render_mode () {
    System.out.println("Render_mode was = "+render_mode);
    if (render_mode.equals("VR"))
        render_mode = "Rel";
    else if (render_mode.equals("Rel"))
        render_mode = "VR";
    set_mode_buttons();
    System.out.println("Render_mode is = "+render_mode);
} // end toggle_render_mode

/**
 * method toggle_potential_mode
 */
void toggle_do_potential () {
    do_potential = !do_potential;
    set_mode_buttons();
} // end toggle_do_potential

} // end Object Mode
```

05/22/00  
23:01:40

## Confidential and Proprietary Property of Rocky Nevin

### Node.java

```
// This is Node.java by Rocky Nevin

import java.lang.*;
import java.util.*;
import java.awt.*;

/*
 * This is Node.java by Rocky Nevin
 * @version 0.4, 3/12/98
 * by Rocky Nevin
 */

// link() now uses a Vector of Link objects
// position_children() returns if TS<=G_Ccurrent_TS and wiggles Root node
// 6/29/99 added Link_CS_R&L, set other node_CS to it in getNodeAllLink()

public class Node extends Object {

    static final int MAX_CHILDREN = 90;
    static final int MAX_TEXT_DATA_LINES = 120;
    static final int MAX_NORMALIZED_MAG = 10;
    static public final double DELTA_MAG = 1.0;
    static public final double MIN_MAG = 0.2;
    static public final double SMALL_MAG = 1.0;
    static public final double MED_MAG = 4.0;
    static public final double BIG_MAG = 7.0;
    static public final double MAX_MAG = 10.0;
    static public final double EMPHASIZED_MAG = 15.0;
    static public final double HYPER_MAG = 20.0;
    static public final double BARELY_VISIBLE_MAG = SMALL_MAG + 0.1;
    static public final double BARELY_INVISIBLE_MAG = SMALL_MAG - 0.1;
    static public final double DEFAULT_MAG = MED_MAG - 0.5;
    static public final double DEBUG_MAG = 20.0;

    boolean anDebug=false;
    boolean anStopSpread=false;
    boolean anIsFrozen=false; // Temporary freezing of position
    boolean anIsPolarized=true; // Bi-directional Link is polarized
    boolean anIsSelected=false;
    boolean isBB=false;
    boolean isMarked=false;
    boolean isFile=false; //
    boolean isDirectory=false; //
    boolean isPN=false; // is Potentiation node
    boolean isCN=false; // is Context node
    boolean isAN=false; // is Abstract node
    boolean isON=false; // is Abstract node
    boolean isDN=false; // is Data node

    boolean isURL=false; // is URL node
    boolean isPOV=false;
    boolean isEvent=false;
    boolean isForm=false;
    boolean isLayer=false;
    boolean there_is_data=false;
    boolean data_access_failed=false;
    boolean there_should_be_data=false;
    double dist = 0;
    double old_dist = 0;
    int Tdist = 0;
    int ANlevel = 0;
    int ID = 0; // Used to track temporary actions on any node
    int ChildNum = 0; // child number of this, counted by its parent (where parent_dist=this_dist-1)
    int ChildCount = 0; // the number of children of this (where child_dist=this_dist+1)
    int BigChildCount = 0; // How many emphasized children there are
    int TransitionCount=0;
    String Type="",
    String Name="",
    String Desc="",
    String Data[];
    // double TinyScale=1.0;
    double ThetaOffset = 0;
    // long LongData;
    long created_TS = 0; // A Time Stamp to record when we last did something to ourself,
    long TS = 0; // A Time Stamp to record when we last did something to ourself,
    long drawnTS = 0; // A Time Stamp to record when we last did something to ourself,
    Vector Links;
    Vector ContextLinks;
    int child_vec[]; // allows us to order fn's called on children
    double px=0, py=0;
    double pxPOV=0, pyPOV=0;
    double pxNeighbors=0, pyNeighbors=0;
    double pxParent=0, pyParent=0;
    Color TempColor;
    static int potentiation=1;
    long potentiation_TS = -1; // A Time Stamp to record when we last were potentiated
    double potentiation_mag = 1;
    double importance = 1;
    // double CS = Link.DEFAULT_CS;
    double mag = DEFAULT_MAG;
    double min_mag = 0;
    long magTS;
    double old_mag = 0.1;
    double delta_mag = 1.0;
    double X=0, Y=0, Z=0;

    // offsets from parent in VRmode in data coordinates
```

05/22/00  
23:01:40

## Confidential and Proprietary Property of Rocky Nevin

### Node.java

2

```
double size_X=6, size_Y=15, dz=1; //
double x=0, y=0, z=0; // temp positions for POVs in data coordinates
double size_x=1, size_y=1, dz=1; // permanent sizes for objects in data coordinates
double mag1=1.0, mag2=1.0, mag3=1.0;
long mag1TS, mag2TS, mag3TS;
//Node VR_node;
boolean V_Ryes;
// static Node Myself;
// double DoubleData;
// DataObj dataobj; // I'm storing data directly into nodes now (5/99)
// Vector GlobalVec;
// static int total_nodes_created = 0;
/**
DS.spreadback_from(target_node, POV)
is a mechanism for finding the shortest route from a POV (or any node) to
any other node. It spreads one level at a time, thus invoked for n-times
until all the return values indicate no more nodes.
It uses TS each invocation, passes desired level to return to n.
The invocation which finds the target returns a code and the caller
modifies the mag to that invocation, spreading backwards directly to the POV
*/
/**
** Node CONSTRUCTORS (constructors)
**
*/
//
public Node() {
    init();
    assign();
}

public Node (String Name) {
    init(); this.Name = Name; //createMyself();
    assign();
}

public Node (String Name, String Type) {
    init(); this.Name = Name; this.setType(Type); //createMyself();
    assign();
}

public Node (String Name, String Type, String Desc) {
    init(); this.Name = Name; this.setType(Type); this.Desc = Desc; //createMyself();
    assign();
}

public Node (String Name, String Type, String Desc, String Data) {
```

```
    init(); this.Name = Name; this.setType(Type); this.Desc = Desc; //createMyself();
    this.Data[0] = Data;
    assign();
}

public Node (String Name, String Type, String Desc,
            int x, int y) {
    init(); this.Name = Name; this.setType(Type); this.Desc = Desc;
    //this.X = x; this.Y = y; //createMyself();
    this.setX(x,y);
    this.Data[0] = Data;
    assign();
}

public Node (String Name, String Type, String Desc,
            int x, int y, int size_x, int size_y) {
    init(); this.Name = Name; this.setType(Type); this.Desc = Desc;
    //this.X = x; this.Y = y;
    this.size_X = size_x; this.size_Y = size_y; //createMyself();
    this.setX(x,y);
    this.Data[0] = Data;
    assign();
}

/**
** Node.setX another place to do all setting of X variables
**
*/
```

```
public void setX (double X, double Y) {
    double oldX=0, oldY=0;

    this.setX("-?- ", X, Y);
    return;
} // end Node.setX

/**
 ** Node.setX another place to do all setting of X variables
 **
 */
public void setX (String CallingFnName, double X, double Y) {
    double oldX=0, oldY=0;

    oldX = this.X;
    oldY = this.Y;
    this.X = X;
    this.Y = Y;

    // HERE WE CAN MONITOR PROBLEM AREAS
    /*****
    if (
        (this.Name.equalsIgnoreCase("TL"))
        || (this.Name.equalsIgnoreCase("Today"))
        || (this.Name.equalsIgnoreCase("2pm:Today"))
        || (this.Name.equalsIgnoreCase("2pm:Tomorrow"))
        || (this.Name.equalsIgnoreCase("2pm:Yesterday"))
        || (this.Name.equalsIgnoreCase("cal"))
        || (this.Name.equalsIgnoreCase("Root"))
    )
    if (oldX==X && oldY==Y)
        System.out.println("Node.setX : "+CallingFnName+
            " "+this.Name+" ".Xvar was ("oldX+", "oldY+") and is unchange
            d");
    else
        System.out.println("Node.setX : "+CallingFnName+
            " "+this.Name+" ".Xvar was ("oldX+", "oldY+") and is changed t
            o -> ("X+", "Y+"));
    *****/
    return;
} // end Node.setX

/**

** set_pot set the potentiation_TS to the argument
**
*/
public void set_pot () {
    this.set_pot(GUL.current_TS);
}

public void set_pot (long pot_TS) {
    this.potentiation_TS = pot_TS;
    // System.out.println("set_pot("+this.Name+"")");
} // end set_pot

/*
 * assign set Node vars, and set Link's vars from Node's
 */
public void assign () {
    Node lnode=null;
    double dx=50*(GUL.random()-0.5), dy=50*(GUL.random()-0.5);

    Link link;
    this.x = this.X + dx;
    this.y = this.Y + dy;
    this.size_x = this.size_X;
    this.size_y = this.size_Y;
    DataSea.add_to_node_vec(this); // this will initialize node_vec if needed
} // end assign

/**
 ** setType
 **
 */
public void setType (String type) {
    int i, size;
    Node tn;

    this.Type = type;

    if (type.equalsIgnoreCase("BB"))
        this.isBB = true;
    if (type.equalsIgnoreCase("PN"))
        this.isPN = true;
    if (type.equalsIgnoreCase("CN"))
        this.isCN = true;
```

05/22/00  
23:01:40

## Confidential and Proprietary Property of Rocky Nevin

### Node.java

```
if (type.equalsIgnoreCase("AN"))
    this.isAN = true;
if (type.equalsIgnoreCase("ON"))
    this.isON = true;
if (type.equalsIgnoreCase("POV"))
    this.isPOV = true;
if (type.equalsIgnoreCase("DN"))
    this.isDN = true;
if (type.equalsIgnoreCase("Event"))
    this.isEvent = true;
if (type.equalsIgnoreCase("Form"))
    this.isForm = true;
if (type.equalsIgnoreCase("Layer"))
    this.isLayer = true;
if (type.equalsIgnoreCase("URL")) {
    this.isURL = true;
}

this.Data[0] = "No Data available for node <"+this.Name+">";

} // end setType

/*
 * ini create a vector called this.Links
 */
public void init () {
    Link link;
    int i;

    this.Links = new Vector(10);
    this.ContextLinks = new Vector(1);
    this.child_wec = new int[MAX_CHILDREN+1];
    this.created_TS = GUI.current_TS;
    this.set_TS();
    this.Data = new String[MAX_TEXT_DATA_LINES];

} // end init

/**
 ** setLinks.should_we_pos
 **
 */
public void setLinks.should_we_pos (Node node, boolean value) {
    Link link;
    if (node == null) {
        System.out.println("Node.setLinks.should_we_pos(): ERROR, node is null");
        return;
    }
}
```

```
link = node.getLinkTo(node);
if (link == null) {
    System.out.println("Node.setLinks.should_we_pos(): ERROR, link is null to "+
        node.Name);
    if (GUI.animation_thread != null)
        GUI.animation_thread.dumpStack();
    return;
}
link.setLinks.should_we_pos(node, value);
} // end setLinks.should_we_pos

/**
 ** setLinks.VRparamsTo
 **
 */
public void setLinks.VRparamsTo (Node node) {
    Link link;
    if (node == null) {
        System.out.println("Node.setLinks.VRparamsTo(): ERROR, node is null");
        return;
    }
    link = node.getLinkTo(node);
    if (link == null) {
        System.out.println("Node.setLinks.VRparamsTo(): ERROR, link is null to "+node.
            de.Name);
        if (GUI.animation_thread != null)
            GUI.animation_thread.dumpStack();
        return;
    }
    link.setLinks.VRparams(node);
} // end setLinks.VRparamsTo

/**
 **
 **
 */
public Node getParent (Node node) {
    Node tnode, saved_node=null;
    int i, size;
    if (node == null)
        return((Node)null);
    size = node.Links.size();
    for (i=0; i<size; i++) {
}
```

05/22/00  
23:01:40

## Confidential and Proprietary Property of Rocky Nevin

### Node.java

```
inode = getNodeAtLink(i);
if (inode==null) {
    break;
}
if (inode.dist == (node.dist - 1)) {
    saved_node = inode;
}
}

return(saved_node);
} // end getParent
```

```
/**
 * method checkSamePol tell if links from parent->this and this->child have the same polarizati
on
 */
public boolean checkSamePol (Node parent, Node child) {
    char parentPol, childPol;
    boolean returned_boolean=false;

    return(true); // SIMPLIFY 11/11/99
```

```
/******
if (!GUI.checkPolarization)
    return(true);
*****
```

```
if (!parent.isPolarized)
    returned_boolean = true;
if (!this.isPolarized)
    returned_boolean = true;
if (!child.isPolarized)
    returned_boolean = true;
```

```
if (returned_boolean == false) {
    parentPol = parent.getPol(this);
    childPol = this.getPol(child);
```

```
if (parentPol == childPol)
    returned_boolean = true;
else
    returned_boolean = false;
}
```

```
if (GUI.Debug == 1)
    System.out.println("checkSamePol(): "+returned_boolean+" between "+parent.Name+"("+parent.
Type+") -> "+this.Name+"("+parent.Type+") -> "+child.Name+"("+parent.Type+")");
return(returned_boolean);
*****
} // end checkSamePol
```

```
/**
** hasSmallerDistThan Returns a sibling which is of Type 'type', else null
**
 */
public boolean hasSmallerDistThan (Node other_node) {
    boolean rel_val;
```

```
if (this.dist < other_node.dist - 0.00001) // handles java imprecision
    rel_val = true;
else
    rel_val = false;
```

```
return(rel_val);
} // end hasSmallerDistThan
```

```
/**
**
** hasLargerDistThan Returns a sibling which is of Type 'type', else null
**
 */
public boolean hasLargerDistThan (Node other_node) {
    boolean rel_val;
```

```
if (this.dist > other_node.dist + 0.00001) // handles java imprecision
    rel_val = true;
else
    rel_val = false;
```

```
return(rel_val);
} // end hasLargerDistThan
```

05/22/00  
23:01:40

## Confidential and Proprietary Property of Rocky Nevin

### Node.java

```
/**
 ** hasSiblingOfType Returns a sibling which is of Type 'type', else null
 **
 */
public Node hasSiblingOfType (String type) {
    int i, size;
    Node child;

    size = this.Links.size();
    for (i=0; i<size; i++) {
        child = (Node)(this.getNodeAtLink(i));
        if (child.Type.equalsIgnoreCase(type))
            return(child);
    }

    return((Node)null);
} // end hasSiblingOfType

/**
 ** goesUpstreamTo
 **
 */
public boolean goesUpstreamTo (Node node) {
    int i, size;
    Node child;

    if ('-' == this.getPol(node))
        return(true);
    else
        return(false);
} // end goesUpstreamTo

/**
 /**
 ** goesDownstreamTo
 **
 */
public boolean goesDownstreamTo (Node node) {
    int i, size;
    Node child;

    if ('+' == this.getPol(node))
        return(true);
    else
        return(false);
} // end goesDownstreamTo

/**
 * method getPol get the polarization between this and node
 */
public char getPol (Node node) {
    char returned_char = '-';

    if (node==null) {
        GUI.WARNING(0, "Node.getNodeAtLink", "node is null");
        return('x');
    }

    Link link = this.getLinkTo(node);
    if (link==null) {
        GUI.WARNING(0, "Node.getNodeAtLink", "link between <"+this.Name
        +"> and <"+node.Name+"> is null");
        return('x');
    }
    if (node == link.NodeR)
        returned_char = '+';
    else
        returned_char = '-';

    return(returned_char);
} // end getPol

/**
 * method getAN_connected_to_DN UN-NECESSARY?
 */
public Node getAN_connected_to_DN (Node DN) {
    int i, j, size;
    Node iAN, iDN;

    size = this.Links.size();
    for (i=0; i<size; i++) {
```

05/22/00  
23:01:40

# Confidential and Proprietary Property of Rocky Nevin

## Node.java

```

    IAN = this.getNodeAtLink(i);
    if (IAN == null)
        return((Node)null);
    j = 0; // set to run through DN's
    if (IAN.isAN()) {
        while (null != (IDN = IAN.getDN(j++)))
        {
            if (IDN == DN)
                return(IAN); // IAN is connected to arg DN
            else
                ; // Keep trying to find the given DN connected to IAN
        }
    }
}

return((Node)null);
} // end getAN_connected_to_DN

/**
 * method getDN_connected_to_AN UN-NECESSARY?
 */
public Node getDN_connected_to_AN (Node AN) {
    int i, j, size;
    Node IDN, IAN;

    size = this.Links.size();
    for (i=0; i<size; i++) {
        IDN = this.getNodeAtLink(i);
        if (IDN == null)
            return((Node)null);
        j = 0; // set to run through AN's
        if (IDN.isDN()) {
            while (null != (IAN = IDN.getAN(j++)))
            {
                if (IAN == AN)
                    return(IDN); // IDN is connected to arg AN
                else
                    ; // Keep trying to find the given AN connected to IDN
            }
        }
    }

    return((Node)null);
} // end getDN_connected to AN

}

/**
 ** getParent
 **
 */
public Node getParent () {
    int size=0, i;
    Node tnode;

    size = this.getChildCount();
    for (i=0; i<size; i++) {
        tnode = this.getCh.Id(i);
        if (tnode.dist < this.dist)
            return(tnode); // parent will have lower dist
    }
    return((Node)null); // return null if we fall through
} // end getParent

/**
 ** getChild
 **
 */
public Node getChild (int i) {
    return(this.getNodeAtLink(i));
} // end getChild

/**
 ** getChildCount
 **
 */
public int getChildCount () {
    return(this.Links.size());
} // end getChildCount

/**
 * method getAN_named return a (remotely) linked AN go dist of 'how_far' max.
 * Later allow return of distance info and such
 */
public Node getAN_named (String name, int how_far) {
    int i, AN_counter=0, size;
    Node tnode=null, ret_node=null;
}
```

05/22/00  
23:01:40

## Confidential and Proprietary Property of Rocky Nevin

Node.java

8

```
if (how_far <= 0) // SINCE WE ARE RECURSIVE, CONSIDER THE END POINT
    return (Node)null; // FAILED TO FIND NODE
```

```
System.out.println("getAN_named(): looking for '"+name+"' from this='"+this.Name+"'");
```

```
size = this.Links.size();
for (i=0; i<size; i++) {
```

```
    inode = this.getNodeAtLink(i);
    System.out.print("getAN_named(): how_far="+how_far+", inode.Name="+inode.
Name+" ");
```

```
    if (inode == null)
        return((Node)null);
    if (inode.dist >= this.dist) {
        System.out.print(" dist="+inode.dist+"(>="+this.dist+" );");
        if (inode.Type.equals("AN")) {
            System.out.print(" is an AN ");
            if (inode.Name.equals(name)) {
                System.out.println(" CORRECT NAME, returning ");
                return(inode);
            }
        }
        else
            System.out.print(" but is the wrong name, != '"+name+"' looping...
```

```
");
```

```
    }
    else
        System.out.print(" not an AN ");
    }
    else
        System.out.print(" dist="+inode.dist+" > "+this.dist);
```

```
System.out.println("");
```

```
// HAVEN'T FOUND CORRECT AN, SO RE-DO LOOP ON CHILDREN AND RECURSE WITH
how_far--
```

```
for (i=0; i<size; i++) {
    inode = this.getNodeAtLink(i);
    if (inode == null)
        return((Node)null);
    if (inode.dist > this.dist) {
        System.out.println("Recurring.");
        rel_node = inode.getAN_named(name, (how_far-1));
        if (rel_node != null)
```

```
        return(rel_node);
    }
    // FALL THROUGH
    return((Node)null);
} // end getAN_named
```

```
/**
 * method getAN return a directly linked AN, skip 'which' of them
 */
public Node getAN (int which) {
    int i, AN_counter=0, size;
    Node inode;
```

```
size = this.Links.size();
for (i=0; i<size; i++) {
    inode = this.getNodeAtLink(i);
    if (inode == null)
        return((Node)null);
    if (inode.Type.equals("AN")) {
        if (AN_counter<which) // if which==0, return first
            AN_counter++;
        else
            return(inode);
    }
}
```

```
return((Node)null);
} // end getAN
```

```
/**
 * method getDN return a directly linked DN, skip 'which' of them
 */
```

```
public Node getDN (int which) {
    int i, DN_counter=0, size;
    Node inode;
```

```
size = this.Links.size();
for (i=0; i<size; i++) {
    inode = this.getNodeAtLink(i);
    if (inode == null)
        return((Node)null);
    if (inode.Type.equals("DN")) {
        if (DN_counter<which) // if which==0, return first
            DN_counter++;
```

```
        else
            return(node);
    }

    return((Node)null);
    // end getDN

/**
 * method getLinkTo
 */
    public Link getLinkTo (Node node) {
        int i, size, which=-1;
        Node mnode;

        size = this.Links.size();
        which = this.isThereLinkTo(node);

        if (which != -1)
            return((Link)(this.Links.elementAt(which))); // got it
        // end getLinkTo

        return((Link)null); // fail-through, default, a failure
    }

/**
 * method getLink
 */
    public Link getLink (int i) {
        return((Link)(this.Links.elementAt(i)));
    }

/**
 * method isThereLinkTo Get the other node, not 'this'
 */
    public int isThereLinkTo (Node node) {
        Link tlink;
        int i, size;

        size = this.Links.size();
        for (i=0; i<size; i++) {
            tlink = (Link)(this.Links.elementAt(i));
            if ((node == tlink.NodeL) || (node == tlink.NodeR))
                return(i);
        }
    }

    return(-1);
    // end isThereLinkTo

/**
 * method set_Desc set the Desc in the correct link to other_node
 */
    public int set_Desc (Node other_node, String desc) {
        if (desc.equals(""))
            return(0);

        Link link = getLinkTo(other_node);
        if (link == null)
            return(-1);

        if (this == link.NodeR) {
            link.Desc.R = desc;
            GUI.P(0, "Node.set_Desc", "Set Desc("+desc+") into link_R of "+this.Name+
                "-> "+other_node.Name);
        }
        else
            if (this == link.NodeL) {
                link.Desc.L = desc;
                GUI.P(0, "Node.set_Desc", "Set Desc("+desc+") into link_L of "+this.Name+
                    "-> "+other_node.Name);
            }
        else {
            GUI.ERROR(0, "Node.set_Desc", "Internal error, this is neither NodeR or NodeL");
            return(-1);
        }
    }

    return(0);
    // end set_Desc

/**
 * method get_Desc get the Desc in the correct link to other_node
 */
    public String get_Desc (Node other_node) {
        String desc=null;

        Link link = getLinkTo(other_node);
        if (link == null)
```

Node.java

```

L");
    GUI.ERROR(0, "Node.set_CS", "Internal error, this is neither NodeR or NodeL");

    return(CS);
} // end set_CS
*/

/**
 * method set_CS_from_CNode Set the CS from this to the other node
 */
public double set_CS_from_CNode (Node left_node, Node right_node, double CS) { // CS

    Link link = left_node.getLinkTo(right_node);
    if (link == null)
        return(0);

    link.set_CS(CS);

    return(CS);
} // end set_CS_from_CNode

/**
 * method get_CS Get the other node, not 'this'
 */
public double get_CS (Node other_node) {

    if (other_node == null)
        return(Link.DEFAULT_CS);

    Link link = getLinkTo(other_node);
    if (link == null)
        return(0);

    if (this == link.NodeR) {
        return(link.CS_L);
    }
    else
        if (this == link.NodeL) {
            return(link.CS_R);
        }
    else
        GUI.ERROR(0, "Node.get_CS", "Internal error, this is neither NodeR or NodeL");

    return(0);
} // end get_CS
}

```

```
/**
 * method getNodeAtLink Get the other node, not 'this'
 */
public Node getNodeAtLink (int i) {
    Link tlink=null;

    if (i >= this.Links.size()) {
        GUI.WARNING(0, "Node.getNodeAtLink",
            "FAILED, i(\"+i+\")>= size of Links vector(\"+this.Links.size()+\")");
        return((Node)null);
    }
    tlink = (Link)(this.Links.elementAt(i));
    if (this == tlink.NodeL) { // decide which node to return
        this.get_CS(tlink.NodeR); // store CS into NodeR, use soon
        return(tlink.NodeR); // return 'other' node
    }
    else {
        this.get_CS(tlink.NodeL); // store CS into NodeL, use soon
        return(tlink.NodeL); // return 'other' node
    }
} // end getNodeAtLink

/**
 * method getCNNodeAtLink Get the CN node
 */
public Node getCNNodeAtLink (int i) { // CS
    Link tlink=null;

    if (i >= this.Links.size()) {
        GUI.WARNING(0, "Node.getCNNodeAtLink",
            "FAILED, i(\"+i+\")>= size of Links vector(\"+this.Links.size()+\")");
        return((Node)null);
    }
    tlink = (Link)(this.Links.elementAt(i));

    return(tlink.CNode);
} // end getCNNodeAtLink

/**
 * method link
 */
public Link link (String name) {
    Node tnode = DataSea.find_node_named(name);
    return(this.link(tnode, "", Link.DEFAULT_CS, false, "polarized"));
} // end of link()

public Link link (Node node, String polarized) {
    return(this.link(node, "", Link.DEFAULT_CS, false, "polarized"));
} // end of link()

//public Link link (Node node, String desc) {
//    return(this.link(node, desc, Link.DEFAULT_CS, false, true));
//} // end of link()

public Link link (Node node, Node CNode) { // link this to node, add CNode to link
    Link link = this.link(node, "", Link.DEFAULT_CS, false, "polarized");
    if (CNode != null)
        link.addCNNode(CNode);
    return(link);
} // end of link()

return(link);
} // end of link()

public Link link (Node node, Node CNode, String polarized) { // link this to node, add CNode to link
    Link link = this.link(node, "", Link.DEFAULT_CS, false, "polarized");
    if (CNode != null)
        link.addCNNode(CNode);
    return(link);
} // end of link()

// MAIN ROUTINE FOR LINK. Called by other overloaded versions
public Link link (Node node, String desc, double CS, boolean OneWay, String polarized) {
    int i=0, size;
    Link tempLink;
    Node tnode;
    boolean isPolarized=true;

    if (polarized != null) {
        if (polarized.equalsIgnoreCase("polarized"))
            isPolarized = true;
    }

    if (node == null) {
        return((Link)null);
    }
    if (node.Links.size() >= MAX_CHILDREN ||
        this.Links.size() >= MAX_CHILDREN) {
```

05/22/00  
23:01:40

# Confidential and Proprietary Property of Rocky Nevin

## Node.java

12

```
        return((Link)null);
    }

    //
    // See if already linked
    size = this.Links.size();
    // this.ChildCount = this.Links.size();
    for (i=0; i<size; i++) {
        tnode = this.getNodeAtLink(i);
        if (tnode == node) {
            return((Link)this.Links.elementAt(i));
        }
    }

    temp_link = new Link();
    temp_link.NodeL = this;
    temp_link.NodeR = node;
    temp_link.isPolarized = isPolarized;
    temp_link.set_CS(CS);
    this.set_Desc(node, desc);
    this.Links.addElement(temp_link);

    if (!OneWay)
        node.Links.addElement(temp_link);

    if (DataSea.currentCNode != null) {
        temp_link.addCNode(DataSea.currentCNode);
        //System.out.print("adding DataSea.currentCNode to link for '<'+this.Name+'>' and '<'+
        node.Name+'>";
    }

    return(temp_link);
} // end of link()

/**
 * method unlink_both Remove both links between this and given node
 */
public void unlink_both (Node node) {
    int i=0, size;
    this.unlink(node);
    node.unlink(this);
    return;
} // end of unlink_both()

/**
 * method unlink Unlink the argument 'node'.
 */
    *
    * Given the node 'node' which is linked to 'this',
    * find the link of 'node' having 'this' as the other half,
    * and then remove that link from 'node'. 'this' is unaffected.
    */
    public void unlink (Node node) {
        int i=0, size;
        Node tnode;
        if (node == null) {
            return;
        }
        size = node.Links.size();
        // System.out.print("unlink from this="+this.Name+", size="+size+" ... node="+
        +node.Name);
        for (i=0; i<size; i++) {
            tnode = node.getNodeAtLink(i);
            // System.out.print(" "+i+" "+node.Name+"");
            if (tnode == this) { // This is the right index
                // System.out.println(" removing link in "+node.Name+"");
                node.Links.removeElementAt(i);
                System.out.println(" removed link from "+this.Name+" -> "+node.N
                ame);
                return; // Early return
            }
        }
        return;
    } // end of unlink()

    /**
    * method unlink_all Unlink all nodes from 'this'.
    * This is probably in preparation for deleting this node.
    */
    public void unlink_all () {
        int i=0, size;
        Node tnode;
        size = this.Links.size();
        for (i=0; i<size; i++) {
            if (size == this.Links.size()) // thread problems????
                tnode = this.getNodeAtLink(i);
            else {
                GUI.WARNING(0, "Node.unlink all",
                "FAILED, size != old size of Links vector");
                return;
            }
        }
        tnode = this.getNodeAtLink(i);
    }

    /**
    * method unlink Unlink the argument 'node'.
    */
}
```

```
if (inode == null) {
    System.out.println("unlink_all: FAILED on inode, null node from thi
s.getNodeAtLink("+i+")");
}
else {
    this.unlink_both(inode);
}
return;
} // end of unlink_all()

/**
 * method undo returns the old mag
 */
double undo (int levels) {
    double prior_mag=0.;
    prior_mag = this.mag;

    if (this.mag1TS >= GUI.lastCommandTS) {
        this.mag = this.mag1;
        this.TS = this.mag1TS;
        this.mag1TS = this.mag2TS;
        this.mag2TS = this.mag3TS;
    }
    if (this.mag1TS >= GUI.lastCommandTS) {
        this.mag = this.mag2;
        this.mag2TS = this.mag3TS;
    }

    if (this.Name.equals("Bob") || this.Name.equals("name"))
        System.out.println("undo: "+this.Name+" .Tdiff="+GUI.current_TS-this.TS
        +", Mags("+this.mag+" "+this.mag1+" "+this.mag2+" "+this.mag3
        +")");
    /*****
    if (levels == 1) {
        this.mag = this.mag1; this.magTS = this.mag1TS;
        this.mag1 = this.mag2; this.mag1TS = this.mag2TS;
        this.mag2 = this.mag3; this.mag2TS = this.mag3TS;
    }
    else
        if (levels == 2) {

this.mag = this.mag2; this.magTS = this.mag2TS;
this.mag1 = this.mag3; this.mag1TS = this.mag3TS;
}

return (prior_mag);
} // end of undo()

/**
 * method set_TS returns the old TS
 */
long set_TS () {
    long old_TS;
    old_TS = this.TS;
    this.TS = GUI.thisCommandTS;
    return(old_TS);
} // end set_TS

/**
 * method TS_diff returns the the millisecond value of (GUI.thisCommandTS - this.TS)
 */
double TS_diff () {
    double diff = GUI.thisCommandTS - this.TS;
    //System.out.println("TS_diff"+this.Name+"="+diff);
    return(diff);
} // end TS_diff

/**
 * method set_theta_offset
 */
void set_theta_offset (double theta_offset) {
    this.set_theta_offset(theta_offset, "Unknown caller");
} // end set_theta_offset

/**
 * method set_theta_offset
 */
}
```

05/22/00  
23:01:40

## Confidential and Proprietary Property of Rocky Nevin

Node.java

14

```
void set_theta_offset (double theta_offset, String str) {
    if (this.Debug && GUI.Debug == 1)
        System.out.println("set_theta_offset(): "+str);
    this.ThetaOffset = theta_offset;
} // end set_theta_offset

/**
 ** calc_new_mag
 **
 */
public double calc_new_mag (double current_mag, double given_new_mag) {
    double temp_mag = Math.exp(current_mag) + given_new_mag;
    if (temp_mag < 2.8)
        temp_mag = 2.8;

    double result_mag = Math.log(temp_mag);

    System.out.println(current_mag+" "+given_new_mag+" -> "+result_mag);
    return(result_mag);
} // end calc_new_mag

/**
 * method set_mag.no.history returns the old mag
 */
double set_mag.no.history (double new_mag) {
    // double temp_new_mag = calc_new_mag(this.mag, new_mag);

    if (new_mag < this.min_mag) // sanity check
        new_mag = this.min_mag;

    if (this.Debug && GUI.Debug == 1)
        System.out.println("set_mag.no.history(): "+this.Name+": mag "+this.mag+" -> "+new_mag);

    if (!GUI.doNormalization) // if we don't normalize, then limit the max value of mag
        new_mag = (new_mag > MAX_MAG) ? MAX_MAG : new_mag;

    this.mag = new_mag;

    return(old_mae);
} // end set_mag.no.history

/**
 * method set_mag with node argument, use CS to node
 */
double set_mag (double new_mag, Node node) {
    Link link = null;
    double temp_mag;

    // double temp_new_mag = calc_new_mag(this.mag, new_mag);

    if (node != null) {
        link = this.getLinkTo(node);
        temp_mag = (link.CS_R + link.CS_L)/2 * new_mag;
        if (this.Debug)
            System.out.println("set_mag of <"+this.Name+"> by node <"+node.Name+">");
    }
    else
        temp_mag = new_mag;

    return(this.set_mag(temp_mag));
} // end two argument version

/**
 * method set_mag returns the old mag
 */
double set_mag (double new_mag) {
    double old_mag = this.mag;
    long diff_TS;

    // double temp_new_mag = calc_new_mag(this.mag, new_mag);

    if (this.Debug)
        System.out.println("set_mag(): "+this.Name+": mag "+this.mag+" -> "+new_mag);

    // Set the elements of the stack of old mags if set_mag
    // has not occurred recently
    if ((this.TS + 1000) < GUI.thisCommandTS) {
        this.mag3 = this.mag2; this.mag3TS = this.mag2TS;
        this.mag2 = this.mag1; this.mag2TS = this.mag1TS;
        this.mag1 = this.mag; this.mag1TS = this.magTS;
        this.maeTS = GUI.thisCommandTS;
    }
}
```

```
    this.set_TSO;

    )

    /*****
    // Do Potentiation calculations ...
    diff_TS = (GUI.current_TS - this.potentiation_TS);
    if (diff_TS < 2000)
        diff_TS = 2000;
    if (diff_TS < 4000) { // -1 <= potentiation <= 1
        // this.potentiation_mag = this.potentiation * 4000/diff_TS;
        this.potentiation_mag = this.potentiation * 2.4; // simpler for now 4/19/2000
        System.out.println("set_mag():      POTENTIATION "
            + "potentiation_mag = " + this.potentiation_mag
            + "mag=" + this.mag
            + " " + this.Name);
    }
    else
        this.potentiation_mag = 1; // else reset it
    //
    /*****
    if (this.Debug)
        System.out.println("set_mag(): "+this.Name+" mag "+this.mag+" -> "+new_mag);

    if (this.potentiation_mag < 0)
        new_mag /= -this.potentiation_mag; //
    else
        new_mag *= this.potentiation_mag; //

    if (new_mag < this.min_mag)
        new_mag = this.min_mag;

    if (!GUI.doNormalization) // if we don't normalize, then limit the max value of mag
        new_mag = (new_mag > MAX_MAG) ? MAX_MAG : new_mag;

    this.mag = new_mag; // necessarily update this.mag

    this.delta_mag = new_mag - this.mag1; // delta dependent on mag1 which is dep' on TS

    return (old_mag);
    } // end of set_mag()

    )

    double lift_to_threshold () {
    double new_mag=this.mag;

    if (new_mag < GUI.relations.threshold+0.1)
        this.set_mag(GUI.relations.threshold+0.1);

    return(new_mag);
    } // end lift_to_threshold

    double lift (double delta) {
    double new_mag=this.mag;

    new_mag += delta;
    if (new_mag >= MAX_MAG)
        new_mag = MAX_MAG;
    this.set_mag(new_mag);
    return(new_mag);
    } // end lift

    double a_little_more_mag () {
    double new_mag=1;
    new_mag = 1.3*this.mag;
    this.set_mag(new_mag);
    return(new_mag);
    } // end a_little_more_mag

    double a_little_less_mag () {
    double new_mag=1;
    new_mag = this.mag/1.1;
    this.set_mag(new_mag);
    return(new_mag);
    } // end a_little_less_mag

    /**
```

05/22/00  
23:01:40

# Confidential and Proprietary Property of Rocky Nevin

## Node.java

16

```
* method more_mag returns the old mag
*/
double more_mag () {
    return(this.more_mag((Node)null));
}

double more_mag (Node node) { // the link is from 'node' to 'this'
    double new_mag=MIN_MAG;
    Link link = null;

    if (node != null) {
        link = this.getLinkTo(node);
        new_mag = (link.CS_R + link.CS_L)/2 + this.mag;
    }
    else
        new_mag = 1 + this.mag;

    // Set a minimum value for this node's mag
    new_mag = (new_mag < BIG_MAG) ? BIG_MAG : new_mag;

    if (this.Debug)
    {
        if (node==null)
            System.out.println("more_mag(): from null node to this="+this.Name+"", new_mag="+n
ew_mag);
        else
            System.out.println("more_mag(): from node="+node.Name+" to this="+this.Name+"",
new_mag="+new_mag);
    }

    return (set_mag(new_mag, node));
} // end of more_mag()

/**
 * method less_mag returns the old mag
 */
double less_mag () {
    return(this.less_mag((Node)null));
}

double less_mag (Node node) { // the link is from 'node' to 'this'
    double new_mag=MIN_MAG;

    new_mag = this.mag/1.2;
    if (new_mag < MIN_MAG)
        new_mag = MIN_MAG;

    return (set_mag(new_mag));
} // end of less_mag()

/**
 * method set_mag returns the new CS
 */
public void set_mag (Node node) {
    Link link = this.getLinkTo(node);
    link.less_CS();

    return;
} // end of set_mag()

/**
 * method set_dist returns the old dist
 */
double set_dist (int new_dist) {
    double tdist, new_double_dist;

    new_double_dist = (double)new_dist;

    return(this.set_dist((double)new_double_dist));
}

double set_dist(double new_dist) {
```

Node.java

Confidential and Proprietary Property of Rocky Nevin

```

double tdist;

/*****
 * *****/
// Thanks to java's imprecision, I need to manipulate double to eliminate 0.00000000002 error
/*****
 * *****/
    if (new_dist > 0) {
        tdist = ((int)(10.0*(new_dist + 0.000001)))/10.0;
        if (tdist != new_dist)
            new_dist = tdist;
    }
/*****
 * *****/

this.old_dist = this.dist;
this.dist = new_dist;

if (new_dist > 0)
    this.importance = this.mag/this.dist;
//this.Tdist = this.dist; // Set this as default
return (this.old_dist);
} // end of set_dist()

/****
 * *****/
    * method set_Tdist
    */
void set_Tdist (int new_Tdist) {
    this.Tdist = new_Tdist;
    return;
}

/****
 * *****/
    * method setData
    */
int setData (String data) {
    this.Data[0] = data;
    return (0);
} // end of setData()

```

---

```

/****
 * *****/
    * method action
    */
    public void action () {
        if (this.Desc.equals("File_mgt"))
            this.file_mgt();
        return;
    } // end of action()

/****
 * *****/
    * Node.file_mgt
    */
    public void file_mgt () {
        GUI.P(0,"Node.file_mgt", "Operating on "+this.Name);
    } // end file_mgt

/****
 * *****/
    * method describe
    */
    public String describe (String ToWhom) {
        if (ToWhom == "To Console") {
            return((String) null);
        }
        if (ToWhom == "To Me") {
            // Default
        }
        return("Name="+this.Name+ "; Type="+this.Type+ ", Desc="+Desc +""");
    } // end of describe()

/****
 * *****/
    *
    */
    public double msg (String action, String msg, long TS, long dt) {
        long deadline_millis;

        deadline_millis = java.lang.System.currentTimeMillis() - (TS + dt);
        // if (deadline_millis > 0)
        GUI.P(0, "Node.msg", "clock expired, " +deadline_millis+ " milliseconds over.");
        // else
        GUI.P(0, "Node.msg", "clock not expired, "
        // +deadline_millis+ " milliseconds over.");
        return(0.0);
    }

```

```
/*
** Try handling various recursions, do something along the network
**/
float network_start (Node node){

    if (node == null)
        return(-1);

    node.dist = 0;

    network_recursive(node);
    return(0);
} // end position_start (a simple function calling recursive fns)

/*
** Try handling various recursions, do something along the network
**/
float network_recursive (Node node){

    if (node == null)
        return(-1);

    node.dist = 0;

    return(0);
} // end position_start (a simple function calling recursive fns)

/*****
public class DataObj extends Object {
    // some elements for data, stubbed now
    public String getDataAsString () { return((String)null); };
} // end class DataObj
*****/

) // End of class Node
```

05/22/00  
23:02:21

# Confidential and Proprietary Property of Rocky Nevin

Force.java

1

```
// This is Force.java by Rocky Nevin

import java.lang.Math;

public class Force extends Object {
    double pxPOV, pyPOV;
    double thetaOffset = 0.1;
    double pxParent, pyParent;
    double pxSiblings, pySiblings;
    double pxNeighbors, pyNeighbors;
    double pxWind, pyWind;
    double pxSum, pySum;
    double ThetaRange=1, theta=1.0, x=0, y=0, dist=0;
    int i;
    double noiseCounter=0, xNoise=0, yNoise=0;
    long local_TS;
    boolean XNeg=false, YNeg=false;
    double opt_dist, alpha, beta;
    static double rand_scale = 10/GUL_mag*scale; // guages severity of heat noise

    static boolean calcPOVPressure=false;
    static boolean calcNeighborPressure=false;
    static boolean calcParentPressure=true;
    static boolean calcNoise=false;

    /**
     * shift_proximally shift proximal nodes by X,Y
     * * This is expensive, called by calc_pressures each time shifted AND/N is found
     */
    public void shift_proximally (Node node, double X, double Y) {
        int i, size;
        Node child;

        if (i ode.dist <= 3) // don't go back too far
            return;

        size = node.links.size();
        for (i=0; i<size; i++) {
            child = (Node)(node.getNodeALink(i));
            if (child.dist < node.dist) {
                child.x = X;
                child.y = Y;
                shift_proximally(child, X, Y);
            }
        }
    }

    } // end shift_proximally

    /**
     * calc_pressures
     * * Put calculated values into force object, apply these to child node
     */
    /**
     * public double calc_pressures (Node parent, Node child) {
     *     double dx, dy, theta, dist;
     *     double delta_x, delta_y, distsq;
     *     double desired_x, desired_y, diff, desired_dist=2, desired_theta=0;
     *     double temp_dist;
     *     int i, size;
     *     int ChildCount=0, ChildNum=0;
     *     int grand_parent, ChildCount=4;
     *     Node inode=null, grand_parent_node;

     *     if ((parent==null) || (child==null)) {
     *         GUL_WARNING(0,"calc_pressures", "NULL: parent="+parent+", child="+child);
     *         return(-1);
     *     }
     *     pxParent=0, pyParent=0;
     *     pxPOV=pyPOV=0;
     *     pxNeighbors=pyNeighbors=0;
     *     pxWind=pyWind=0;
     *     pxSum=pySum=0;

     *     if (child.isFrozen)
     *         return(0);

     *     /*****
     *     // This is expensive, called by calc_pressures each time shifted AND/N is found
     *     if (!child.isLayer)
     *         if (GUL_VRyes(child)) {
     *             child.x = child.X;
     *             child.y = child.Y;
     *             shift_proximally(child, child.X, child.Y);
     *         }
     *     /*****
     */
}
```

05/22/00  
23:02:21

# Confidential and Proprietary Property of Rocky Nevin

Force.java

2

```

if (child.mag < GUI.pos.threshold)
    return(0);
if (child.mag <= 0.2)
    return(0);

/*****
** // BEGIN WIND PRESSURE BACK TO DATAEA
**   if (child.y<0)
**       pyWind=1;
**   pyWind=1;
**   *****/
if (GUI.drawWind && child.BigChildCount>2)
    pyWind = child.BigChildCount * 3.0;

if (GUI.mode_obj.position_mode.equals("Grid"))
{
    temp_dist = (double)(child.dist);
    if (temp_dist <= 0.0)
        temp_dist = 1.0;
    desired_x = 150-50*(temp_dist); // compresses towards the left
    desired_y = -100 + 10*child.mag;
    pxPOV = (desired_x - child.x);
    pyPOV = (desired_y - child.y);
}
else
    if (GUI.mode_obj.position_mode.equals("Lvls"))
    {
        /*****
        *   delta_y = GUI.datasea.POV.y - child.y;
        *   dist = Math.sqrt(delta_y * delta_y);
        *   alpha = 50;
        *   opt_dist = alpha * child.dist;
        *   desired_x = parent.x + 10*alpha*(child.ChildNum-1)-0.5*parent.ChildCount)/(double)(p
        *   arent.ChildCount * child.dist);
        *   desired_y = GUI.datasea.POV.y + opt_dist;
        *   pxPOV = desired_x - child.x;
        *   pyPOV = desired_y - child.y;
        *   *****/
        // NEW THING ... Organize nodes by Node.dist, URLs by Node.mag separated into two half-fiel
        ds
        if (!child.isAN) { // is not an AN
            desired_x = 20+10*(double)(Node.MAX_MAG - child.mag); // compresses towards the
            right
        } else { // is AN
            //temp_dist = (double)(child.Tdist); // - 1;
            temp_dist = (double)(child.AN.level);
        }
    }
}

if (temp_dist <= 0.0)
    temp_dist = 1.0;
desired_x = -50*(temp_dist); // compresses towards the left
}
desired_y = parent.y + 10*child.ChildNum;
pxPOV = (desired_x - child.x);
pyPOV = (desired_y - child.y);
} // END NEW LEVEL POSITIONING

else if (GUI.mode_obj.position_mode.equals("Rel"))
{ // START RELATIVE POSITIONING

// BEGIN POV PRESSURE // Optimal dist to POV is 50+300/child.mag
if (child.mag > 1) { // DON'T BOTHER IF TOO SMALL, THIS IS AN EXPENSIVE OPERATI
ON
    if ((calcPOVPressure==true) && (GUI.datasea.POV != null)) {
        delta_x = GUI.datasea.POV.x - child.x;
        delta_y = GUI.datasea.POV.y - child.y;
        theta = GUI.get_angle(delta_x, delta_y);
        distsq = (delta_x*delta_x)+(delta_y*delta_y);
        dist = Math.sqrt(distsq);
        pxPOV = (dist-50-300/child.mag) * .1 * Math.cos(theta);
        pyPOV = (delta_y+50+300/child.mag); // * Math.sin(theta);
    }
} // end POV PRESSURE

// BEGIN PARENT PRESSURE
if (calcParentPressure==true) {
    grand_parent_node = parent.getParent();
    if (grand_parent_node != null)
        grand_parent_ChildCount = grand_parent_node.ChildCount;
    else
        grand_parent_ChildCount = 4;

    if (parent == GUI.datasea.POV)
        desired_dist = 10;
    else
        desired_dist = parent.mag * 40/parent.dist; // 7/26/99
    if (child.isAN)
        desired_dist += desired_dist; // double dist for ANs
    ChildCount = parent.ChildCount; // 6/8/99
    ChildNum = child.ChildNum; // 6/8/99
    ThetaRange = GUI.spread_factor/(double)(grand_parent_ChildCount);
    desired_theta = ThetaRange*ChildNum + parent.ThetaOffset;
}
}

```

05/22/00  
23:02:21

## Confidential and Proprietary Property of Rocky Nevin

Force.java

```
desired_theta -= 0.2;
desired_theta = GUI.spread_factor*(ChildNum - ChildCount/2) + parent.ThetaOffset;
//desired_theta += (3.0/child.mag)*thetaOffset; // make spirals 6/21/99

if (parent.isURL) { // 8/24/99
    desired_dist *= child.mag/4.0;
}
else
    if (parent.isBB) { // 8/24/99
        desired_dist *= child.mag/4.0;
    }
    else
        if (parent.isDN) {
            desired_dist *= child.mag/25.0;
        }
        child.ThetaOffset=desired_theta;
        //desired_x = parent.x + desired_dist * Math.sin(desired_theta); // 6/8/99
        //desired_y = parent.y + desired_dist * Math.cos(desired_theta); // 6/8/99
        desired_x = parent.x + desired_dist * Math.cos(desired_theta); // 6/8/99
        desired_y = parent.y + desired_dist * Math.sin(desired_theta); // 6/8/99
        pxParent = 0.5*(desired_x - child.x); // 6/8/99
        pyParent = 0.5*(desired_y - child.y); // 6/8/99
    }
}

/*****
if (child.isDN) {
}
*****/

} // END RELATIVE POSITIONING

// BEGIN NEIGHBOR PRESSURE
    if (calcNeighborPressure==true) {
        // DON'T BOTHER IF TOO SMALL, THIS IS AN EXPENSIVE OPERATION
        if (((child.mag >= GUI.pos.threshold) && (child.isLayer)) {
            {
                size = GUI.datasea.node.vec.size();
                for (i=0; i<size; i++) {
                    tnode = (Node)GUI.datasea.node.vec.elementAt(i);
                    if (((tnode != child)
                        && ((tnode.isAN && child.isAN)
                            || ((tnode.isAN && !child.isAN))) {
                        //System.out.println(tnode.Name+" ");
                    }
                }
            }
        }
    }

    delta_x = tnode.x - child.x;
    delta_y = tnode.y - child.y;
    distsq = (delta_x*delta_x) + (delta_y*delta_y);
    desired_dist = 3*(tnode.size.y + child.size.y);
    if (distsq < (desired_dist*desired_dist)) {
        theta = GUI.get_angle(delta_x, delta_y);
        dist = Math.sqrt(distsq);
        if (!GUI.mode.obj.position.mode.equals("Lvis"))
            pxNeighbors -= (desired_dist-dist)*Math.cos(theta);
            pyNeighbors -= 3*(desired_dist-dist)*Math.sin(theta);
        }
    }
}

// maybe_noise();

pxSum = pxParent + pxPOV + pxNeighbors + pxWind + xNoise;
pySum = pyParent + pyPOV + pyNeighbors + pyWind + yNoise;
if (pxSum < 0)
    pxSum = -Math.sqrt(-pxSum);
else
    pxSum = Math.sqrt(pxSum);
if (pySum < 0)
    pySum = -Math.sqrt(-pySum);
else
    pySum = Math.sqrt(pySum);

if (pxSum > GUI.globalMaxPressure)
    GUI.globalMaxPressure = pxSum;
if (pySum > GUI.globalMaxPressure)
    GUI.globalMaxPressure = pySum;

return(0);
} // End of Force.calc.pressures

/**
** noise added to image to avoid stable points and show aliveness
*/
public void maybe_noise () {
    if (calcNoise) {
        xNoise=0;
        yNoise=0;
        return;
    }
}
```

05/22/00  
23:02:21

Confidential and Proprietary Property of Rocky Nevin

Force.java

4

```
    }  
    if (noiseCounter==0) {  
        xNoise = rand_scale*(GUI.random()-.5);  
        yNoise = rand_scale*(GUI.random()-.5);  
    }  
    else {  
        if (noiseCounter>11)  
            noiseCounter=-1;  
        xNoise=0;  
        yNoise=0;  
    }  
    noiseCounter ++;  
  
    xNoise = rand_scale*(GUI.random()-.5);  
    yNoise = rand_scale*(GUI.random()-.5);  
    } // end noise  
} // end of Object Force
```

05/22/00  
23:02:26

# Confidential and Proprietary Property of Rocky Nevin

## GetURLInfo.java

1

```
// This is GetURLInfo.java by Rocky Nevin
// Started on Flanagan, heavily modified
```

```
// This example is from the book "Java in a Nutshell" by David Flanagan.
// Written by David Flanagan. Copyright (c) 1996 O'Reilly & Associates.
// You may study, use, modify, and distribute this example for any purpose.
// This example is provided WITHOUT WARRANTY either expressed or implied.
```

```
import java.net.*;
import java.io.*;
import java.util.*;
```

```
// Return an array of Strings holding the data
// Parse the path minus the file name, use that to follow hyperlinks
// Create nodes for each URL, link them
```

```
public class GetURLInfo {
    static String str_array[];
    static int MAX_LINES = Node.MAX_TEXT_DATA_LINES;
```

```
    public static void printInfo(URLConnection u) throws IOException {
        int index=0, end=0;
```

```
        // Display the URL address, and information about it.
        if (u == null) {
```

```
            System.out.println("GetURLInfo.printInfo(): connection is null");
            return;
        }
```

```
    else { }
```

```
    /*****
    System.out.println("GetURLInfo.printInfo(): connection is "<"+u+">");
```

```
    System.out.println(" Content Type: " + u.getContentType());
```

```
    System.out.println(" Content Length: " + u.getContentLength());
```

```
    System.out.println(" Last Modified: " + new Date(u.getDateAsModified()));
```

```
    System.out.println(" Expiration: " + u.getExpiration());
```

```
    System.out.println(" Content Encoding: " + u.getContentEncoding());
```

```
    System.out.println(" getHeaderField(0): "+u.getHeaderField(0));
```

```
    System.out.println(" getHeaderField(1): "+u.getHeaderField(1));
```

```
    System.out.println(" getHeaderField(2): "+u.getHeaderField(2));
```

```
    System.out.println(" getHeaderField(3): "+u.getHeaderField(3));
```

```
    System.out.println(" getHeaderField(4): "+u.getHeaderField(4));
```

```
    System.out.println(" getHeaderField(5): "+u.getHeaderField(5));
```

```
    System.out.println(" getHeaderField(6): "+u.getHeaderField(6));
```

```
    System.out.println(" getHeaderField(7): "+u.getHeaderField(7));
```

```
    System.out.println(" getHeaderField(8): "+u.getHeaderField(8));
```

```
*****
```

```
// Read and print out the first MAX_LINES lines of the URL.
```

```
System.out.println("First "+MAX_LINES+" lines:");
```

```
DataInputStream in = new DataInputStream(u.getInputStream());
```

```
for(int i = 0; i < MAX_LINES; i++) {
```

```
    String line = in.readLine();
```

```
    str_array[i] = line;
```

```
    if (line == null) break;
```

```
    /*****
    index = line.toLowerCase().indexOf("href");
```

```
    if (index > 0) {
```

```
        index += 6;
```

```
        end = line.toLowerCase().indexOf(" ", index+1);
```

```
        if (end>index && end < line.length())
```

```
            System.out.println("====> URL? <" + line.substring(index, end)
```

```
+ ">");
```

```
    else
```

```
        System.out.println("====> PROBLEM: index="+index+", end="
```

```
+end);
```

```
    }
```

```
    System.out.println("i="+i+" "+line);
```

```
    /*****
    /*****/
```

```
    }
```

```
// Create a URL from the specified address, open a connection to it,
```

```
// and then display information about the URL.
```

```
public String[] getURL(String name, int max_lines)
```

```
throws MalformedURLException, IOException
```

```
{
```

```
    MAX_LINES = max_lines; // OVERRIDE THE DEFAULT HERE
```

```
    System.out.println("get URL: MAX_LINES is now "+MAX_LINES);
```

```
    return(get_URL(name));
```

```
    } // end of max_lines version of get_URL()
```

```
    public String[] getURL(String name)
```

```
throws MalformedURLException, IOException
```

```
{
```

```
    str_array = new String[MAX_LINES];
```

```
    URL url = new URL(name);
```

```
    URLConnection connection = url.openConnection();
```

```
    System.out.println("GetURLInfo.getURL(): getHost(): "+url.getHost());
```

```
    System.out.println("GetURLInfo.getURL(): getRef(): "+url.getRef());
```

```
    System.out.println("GetURLInfo.getURL(): toString(): "+url.toString());
```

05/22/00  
23:02:26

Confidential and Proprietary Property of Rocky Nevin

GetURLInfo.java

2

```
if ((0 < name.toLowerCase().indexOf("http")) && !GUI.NeIOK) {  
    System.out.println("GetURLInfo.getURL(): GUI.NeIOK is false, aborting data retrieval.  
");  
    return(str_array);  
}  
printlnf(connection);  
return(str_array);  
}
```

05/23/00  
00:16:51

## Confidential and Proprietary Property of Rocky Nevin

### Input.java

```
// This is Input.java

import java.io.*;
import java.util.*;
import java.lang.*;

public class Input extends Object {
    GUI gui; // The GUI passed to the Input constructor
}
```

```
public Input (GUI passed_gui_object) {    // Constructor
    gui = passed_gui_object;
}
```

```
/*
 *
 *
 */
```

```
// This method breaks a specified label up into an array of words.
// It uses the StringTokenizer utility class.
// From Nutschell, chapter 5, Multit.java
public Node string_input (String input_string) {
    int num_words;
    String words[];
    String tempStr;
    Node new_node, rel_node=null;
    double event_offset = 0;
    String event_string="";
}
```

```
GUI.P(2, "data:ea.string_input", "<"+input_string+">");
if (input_string.equals("")) {
    tempStr = GUI.priorCommand;
    GUI.priorCommand = "";
    return (rel_node=string_input(tempStr)); // re--invoke
}
GUI.priorCommand = input_string; // for use above, later
```

// StringTokenizer can handle converting strings to numbers, but can't count tokens  
// StringTokenizer can not convert but can count.  
// The number result from StringTokenizer can't easily be converted back into a string.  
// Its all unbelievably stupid and complex.

```
// StringReader r = new StringReader(input_string);
// StringTokenizer t = new StringTokenizer(r);
```

```
StringTokenizer st = new StringTokenizer(input_string, " \t\n\r");
```

```
num_words = st.countTokens();
words = new String[num_words];
```

```
event_offset = 0;
// accumulated_event_offset = 0;
for(int i = 0; i < num_words; i++) {
    words[i] = st.nextToken();
}
```

```
GUI.lastCommandTS = GUI.thisCommandTS;
GUI.thisCommandTS = GUI.currentTS;
rel_node = word_selector(words, input_string, num_words);
```

```
gui.data:ea.normalize();
```

```
return (rel_node);
} // end string_input
```

```
public Node word_selector (String[] words, String input_string, int num_words) {
    Node new_node, rel_node=null;
    String command;
    boolean understood = true;
}
```

```
if (num_words >= 1)
    command = words[0];
else {
    return (rel_node);
}
```

```
GUI.P(1, "word_selector", "num_words="+num_words+", input="+input_string);
```

```
if (num_words == 0)
    return (rel_node);
else
    if (command == null)
        return (rel_node);
```

```
GUI.P(1, "data:ea.word_selector", input_string);
```

// COMMANDS Commands commands

```
if (command.equals("net")) { //
    gui.NetOK = !gui.NetOK;
    gui.P(0, "word_selector", "NetOK = "+gui.NetOK);
}
else
    if (command.equals("t0")) { //
        gui.max transition count = 0;
    }
```

05/23/00  
00:16:51

# Confidential and Proprietary Property of Rocky Nevin

## Input.java

2

```
ion_count);
    }
    else
        if (command.equals("t1")) { //
            gui.max_transition_count = 1;
            GUI.P(0,"datasea.word_selector", "gui.max_transition_count="+gui.max_transit
ion_count);
        }
        else
            if (command.equals("t2")) { //
                gui.max_transition_count = 2;
                GUI.P(0,"datasea.word_selector", "gui.max_transition_count="+gui.max_transit
ion_count);
            }
            else
                if (command.equals("t3")) { //
                    gui.max_transition_count = 3;
                    GUI.P(0,"datasea.word_selector", "gui.max_transition_count="+gui.max_transit
ion_count);
                }
                else
                    if (command.equals("t4")) { //
                        gui.max_transition_count = 4;
                        GUI.P(0,"datasea.word_selector", "gui.max_transition_count="+gui.max_transit
ion_count);
                    }
                    else
                        if (command.equals("t5")) { //
                            gui.max_transition_count = 5;
                            GUI.P(0,"datasea.word_selector", "gui.max_transition_count="+gui.max_transit
ion_count);
                        }
                        else
                            if (command.equals("t6")) { //
                                gui.max_transition_count = 6;
                                GUI.P(0,"datasea.word_selector", "gui.max_transition_count="+gui.max_transit
ion_count);
                            }
                            else
                                if (command.equals("t7")) { //
                                    gui.max_transition_count = 7;
                                    GUI.P(0,"datasea.word_selector", "gui.max_transition_count="+gui.max_transit
ion_count);
                                }
                                else
```

```
        if (command.equals("g1")) { //
            gui.datasea.group(gui.datasea.POV,1);
        }
        else
            if (command.equals("g2")) { //
                gui.datasea.group(gui.datasea.POV,2);
            }
            else
                if (command.equals("g3")) { //
                    gui.datasea.group(gui.datasea.POV,3);
                }
                else
                    if (command.equals("g4")) { //
                        gui.datasea.group(gui.datasea.POV,4);
                    }
                    else
                        if (command.equals("max")) { // find maximum mag node
                            gui.datasea.find_max();
                        }
                        else
                            if (command.equals("rel")) { // release a node (its link) from the POV
                                gui.datasea.word_release(words, num_words);
                            }
                            else
                                if (command.equals("freeze")) { //
                                    gui.datasea.freeze();
                                }
                                else
                                    if (command.equals("unfreeze")) { //
                                        gui.datasea.unfreeze();
                                    }
                                    else
                                        if (command.equals("flip")) { //
                                            gui.FlipAxes = !gui.FlipAxes;
                                        }
                                        else
                                            if (command.equals("bound")) { //
                                                gui.showBoundaries;
                                                GUI.P(0,"datasea.word_selector", "showBoundaries is "+gui.showBoundaries);
                                                if (gui.showBoundaries)
                                                    gui.datasea.set_Tdist_start(gui.lastNode);
                                            }
                                            else
                                                if (command.equals("norm")) { // shall we normalize?
                                                    gui.doNormalization = !gui.doNormalization;
                                                    gui.status("doNormalization is "+gui.doNormalization);
                                                }
```

-108-

4

```

// else
// if (command.equals("id")) { // lines distal
//     word_mode.command(words, num_words);
// }
//     else
// if (command.equals("ip")) { // lines proximal
//     word_mode.command(words, num_words);
// }
// }
// *****/
//     else
// if (command.equals("sr")) { // spread radial
//     word_mode.command(words, num_words);
// }
//     else
// if (command.equals("sd")) { // spread distal
//     word_mode.command(words, num_words);
// }
//     else
// if (command.equals("sp")) { // spread proximal
//     word_mode.command(words, num_words);
// }
//     else
// if (command.equals("upurs")) {
//     gui.datasea.upurs(words, num_words);
// }
//     else
// if (command.equals("shiftOut")) {
//     gui.datasea.shiftOut(words, num_words);
// }
//     else
// if (command.equals("shiftIn")) {
//     gui.datasea.shiftIn(words, num_words);
// }
//     else
// if (command.equals("shiftOutAll")) {
//     gui.datasea.shiftOutAll(words, num_words);
// }
//     else
// if (command.equals("shiftOut")) {
//     gui.datasea.shiftOut(words, num_words);
// }
//     else

```

05/23/00  
00:16:51

Input.java

5

```
if (command.equals("heavy1")) {
    gui.datasea.heavyans((Node)null, gui.lastNode, 1, 0);
}
else
    if (command.equals("heavy2")) {
        gui.datasea.heavyans((Node)null, gui.lastNode, 2, 0);
    }
    else if (command.equals("heavy3")) {
        gui.datasea.heavyans((Node)null, gui.lastNode, 3, 0);
    }
    else if (command.equals("heavy4")) {
        gui.datasea.heavyans((Node)null, gui.lastNode, 4, 0);
    }
    else if (command.equals("an1")) {
        gui.datasea.mag.ans((Node)null, gui.lastNode, 1, 0);
    }
    else if (command.equals("an2")) {
        gui.datasea.mag.ans((Node)null, gui.lastNode, 2, 0);
    }
    else if (command.equals("an3")) {
        gui.datasea.mag.ans((Node)null, gui.lastNode, 3, 0);
    }
    else if (command.equals("an4")) {
        gui.datasea.mag.ans((Node)null, gui.lastNode, 4, 0);
    }
    else
        if (command.equals("ans")) {
            gui.datasea.ans();
        }
        else
            if (command.equals("storebig")) {
                gui.datasea.storebig();
            }
            else
                if (command.equals("x1")) {
                    gui.datasea.storebig();
                }
                else
                    if (command.equals("x2")) {
                        gui.datasea.flattenANS();
                    }
                    else
                        if (command.equals("x3")) {
                            gui.datasea.absURLs();
                        }
                        else
                            if (command.equals("x4")) {
                                gui.datasea.inhstored();
                            }
                            else
                                if (command.equals("connect")) {
                                    gui.datasea.connect_all_to_POVO();
                                }
                                else
                                    if (command.equals("absURLs")) {
                                        gui.datasea.absURLs();
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

```

    }
    else
        if (command.equals("inhstored")) {
            gui.datasea.inhstored();
        }
        else
            if (command.equals("xabs")) {
                gui.datasea.storebig();
                gui.datasea.flattenANS();
                gui.datasea.absURLs();
                gui.datasea.inhstored();
            }
            else
                if (command.equals("selectrecent")) {
                    gui.datasea.selectrecent();
                }
                else
                    if (command.equals("flattenANS")) {
                        gui.datasea.flattenANS();
                    }
                    else
                        if (command.equals("abs")) {
                            gui.datasea.abs(words, num_words);
                        }
                        else
                            if (command.equals("abs1")) {
                                gui.datasea.abs(words, num_words);
                            }
                            else
                                if (command.equals("abs2")) {
                                    gui.datasea.abs(words, num_words);
                                }
                                else
                                    if (command.equals("abs3")) {
                                        gui.datasea.abs(words, num_words);
                                    }
                                    else
                                        if (command.equals("sim")) {
                                            gui.datasea.sim(words, num_words, '+');
                                        }
                                        else
                                            if (command.equals("unsim")) {
                                                gui.datasea.sim(words, num_words, '-');
                                            }
                                            else
                                                if (command.equals("pump")) {
                                                    // pump up all distal nodes
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

05/23/00  
00:16:51

## Confidential and Proprietary Property of Rocky Nevin

Input.java

6

```
        gui.datasea.pump(gui.lastNode);
    }
    else
        if (command.equals("local")) {
            gui.datasea.local(gui.lastNode);
        }
    else
        if (command.equals("chain")) { // mag similar nodes distally
            gui.datasea.chain(gui.lastNode);
        }
    else
        if (command.equals("sides")) { // strip off AN's, see sides of data
            gui.datasea.sides(gui.lastNode);
        }
    else
        if (command.equals("org")) { // Organize children's theta position by mag
            gui.datasea.theta_org = !gui.datasea.theta_org;
            gui.status("theta_org = "+gui.datasea.theta_org);
            GUI.P(0, "word_selector", "theta_org = "+gui.datasea.theta_org);
        }
    else
        if (command.equals("strip")) { // strip off DN's, see only AN's
            gui.datasea.strip(words, num_words);
        }
    else
        if (command.equals("and")) {
            gui.datasea.and(words, num_words);
        }
    else
        if (command.equals("backup")) {
            gui.datasea.backup();
        }
    else
        if (command.equals("pot")) {
            gui.datasea.pot(words, num_words);
        }
    else
        if (command.equals("potmag")) {
            Node.potentiation = 1;
            gui.datasea.potmag(words, num_words);
        }
    else
        if (command.equals("npotmag")) {
            Node.potentiation = -1;
            gui.datasea.potmag(words, num_words);
        }

        else
            if (command.equals("cats")) {
                gui.datasea.cats(words, num_words);
            }
        else
            if (command.equals("backs")) {
                gui.datasea.backs(words, num_words);
            }
        /*****
        else
            if (command.equals("upstream")) {
                gui.datasea.upstream(words, num_words);
            }
        *****/
        else
            if (command.equals("back3")) {
                gui.datasea.back3(words, num_words);
            }
        else
            if (command.equals("back")) {
                gui.datasea.back(words, num_words);
            }
        else
            if (command.equals("Back")) {
                gui.datasea.Back(words, num_words);
            }
        else
            if (command.equals("dumpCNs")) {
                gui.datasea.dumpCNs();
            }
        else
            if (command.equals("showCNs")) {
                gui.datasea.showCNs();
            }
        else
            if (command.equals("showevents")) {
                gui.datasea.showevents();
            }
        else
            if (command.equals("showtime")) {
                gui.datasea.showtime();
            }
    }
```

## Input.java

```
else
    if (command.equals("link")) { //
        gui.datasea.word_link(words, num, words);
        gui.datasea.needitUpdate = true;
    }
    else
        if (command.equals("unlink")) { //
            gui.datasea.word_unlink(words, num, words);
            gui.datasea.needitUpdate = true;
        }
    else
        if (command.equals("delete")) { //
            gui.datasea.word_delete(words, num, words);
            gui.datasea.needitUpdate = true;
        }
    else
        if (command.equals("show")) {
            gui.datasea.show(input_string);
        }
    else
        if (command.equals("setPOV")) {
            gui.datasea.set_POV();
            gui.datasea.needitUpdate = true;
        }
    else
        if (command.equals("one")) {
            gui.datasea.reset_mags();
        }
    else
        if (command.equals("istupdate")) {
            gui.datasea.needitUpdate = true;
        }
    else
        if (command.equals("quick")) {
            gui.quick = !gui.quick;
            gui.P(0, "word_selector", "quick" = "+gui.quick);
            gui.status("quick" = "+gui.quick);
        }
    else
        if (command.equals("resetprint")) {
            GUI_global_str_size = 0;
        }
    else
        if (command.equals("resetCS")) {
            gui.datasea.reset_CSs();
        }
}

else
    if (command.equals("save")) {
        gui.datasea.word_save();
    }
    else
        if (command.equals("reset")) {
            gui.datasea.word_reset();
            gui.datasea.needitUpdate = true;
        }
    else
        if (command.equals("X")) {
            gui.set_Xnode();
        }
    else
        if (command.equals("r")) {
            gui.datasea.word_reset();
            gui.auto_rescale = true;
            gui.status("auto_rescale" = "+gui.auto_rescale);
        }
    else
        if (command.equals("buttons")) {
            gui.show_buttons();
        }
    else
        if (command.equals("lines")) {
            gui.mode_obj.toggle_lines_mode();
        }
    else
        if (command.equals("enh")) {
            gui.datasea.enhance(words, num, words);
        }
    else
        if (command.equals("m1")) {
            gui.datasea.mag(words, num, words, "both", 1, "+");
        }
    else
        if (command.equals("m2")) {
            gui.datasea.mag(words, num, words, "both", 2, "+");
        }
    else
        if (command.equals("m3")) {
            gui.datasea.mag(words, num, words, "both", 3, "+");
        }
    else
        if (command.equals("m4")) {
            gui.datasea.mae(words, num, words, "both", 4, "+");
        }
}
```

```
    }
    else
        if (command.equals("m5")) {
            gui.datasea.mag(words, num_words, "both", 5, "+");
        }
        else
            if (command.equals("m6")) {
                gui.datasea.mag(words, num_words, "both", 6, "+");
            }
            else
                if (command.equals("m7")) {
                    gui.datasea.mag(words, num_words, "both", 7, "+");
                }
                else
                    if (command.equals("magdown")) {
                        gui.datasea.magdownstream(words, num_words);
                    }
                    else
                        if (command.equals("mag")) {
                            gui.datasea.mag(words, num_words, "both", 3, "+");
                        }
                        else
                            if (command.equals("choices")) { // print ANs > BIG.MAG for each dist from Thea
                                gui.datasea.choices();
                            }
                            else
                                if (command.equals("magtype")) { // mag all distally to infinity
                                    gui.datasea.magtype(gui.lastNode, (String) null, 'd');
                                    gui.datasea.magtype(gui.lastNode, (String) null, 'p');
                                }
                                else
                                    if (command.equals("magd")) { // mag all distally to infinity
                                        gui.datasea.magd();
                                    }
                                    else
                                        if (command.equals("Mag")) {
                                            gui.datasea.mag(words, num_words, "both", 6, "+");
                                        }
                                        else
                                            if (command.equals("d1")) {
                                                gui.datasea.mag(words, num_words, "both", 1, "-");
                                            }
                                            else
                                                if (command.equals("d2")) {
                                                    gui.datasea.mag(words, num_words, "both", 2, "-");
                                                }
                                                else
                                                    if (command.equals("d3")) {
                                                        gui.datasea.mag(words, num_words, "both", 3, "-");
                                                    }
                                                    else
                                                        if (command.equals("d4")) {
                                                            gui.datasea.mag(words, num_words, "both", 4, "-");
                                                        }
                                                        else
                                                            if (command.equals("d5")) {
                                                                gui.datasea.mag(words, num_words, "both", 5, "-");
                                                            }
                                                            else
                                                                if (command.equals("d6")) {
                                                                    gui.datasea.mag(words, num_words, "both", 6, "-");
                                                                }
                                                                else
                                                                    if (command.equals("d7")) {
                                                                        gui.datasea.mag(words, num_words, "both", 7, "-");
                                                                    }
                                                                    else
                                                                        if (command.equals("dec")) {
                                                                            gui.datasea.mag(words, num_words, "both", 3, "-");
                                                                        }
                                                                        else
                                                                            if (command.equals("Dec")) {
                                                                                gui.datasea.mag(words, num_words, "both", 5, "-");
                                                                            }
                                                                            else
                                                                                if (command.equals("input")) {
                                                                                    rel_node = words.input(words, input_string, num_words);
                                                                                }
                                                                                else
                                                                                    if (command.equals("zoom")) {
                                                                                        gui.datasea.word_zoom(words, num_words);
                                                                                        gui.magscale *= 3.0;
                                                                                        gui.auto_rescale = false;
                                                                                        gui.status("auto_rescale = "+gui.auto_rescale);
                                                                                    }
                                                                                    else
                                                                                        if (command.equals("a")) {
                                                                                            gui.spread_factor -= 0.05;
                                                                                            GUI.P(0, "word_selector", "spread_factor = "+gui.spread_factor);
                                                                                        }
                                                                                        else
                                                                                            if (command.equals("A")) {
```

```
gui.spread_factor += 0.05;
GUI.P(0,"word_selector","spread_factor="+gui.spread_factor);
}
else
if (command.equals("d")) {
gui.datasea.word_dump(words, num_words);
}
else
if (command.equals("D")) {
gui.datasea.word_dump(words, num_words);
}
else
if (command.equals("tree")) {
gui.print_tree(words, num_words);
}
else
if (command.equals("printup")) {
gui.datasea.print_upstream((Node)null, gui.lastNode);
}
else
if (command.equals("print")) { // print results of all
gui.datasea.print();
}
else
if (command.equals("oldprint")) {
gui.datasea.word_print(words, num_words);
}
else
if (command.equals("state")) {
gui.state();
}
else
if (command.equals("halt")) {
gui.request_stop_thread();
}
else
if (command.equals("run")) {
gui.run_thread();
}
else
if (command.equals("stop")) {
gui.stop_thread();
}
else
if (command.equals("scale")) {
gui.auto_rescale = !gui.auto_rescale;
```

---

```
GUI.P(0,"word_selector","auto_rescale="+gui.auto_rescale);
gui.status("auto_rescale="+gui.auto_rescale);
}
else
if (command.equals("in")) {
gui.magscale *= 2.0;
gui.auto_rescale = false;
gui.status("auto_rescale="+gui.auto_rescale);
}
else
if (command.equals("In")) {
gui.magscale *= 6.0;
gui.auto_rescale = false;
gui.status("auto_rescale="+gui.auto_rescale);
}
else
if (command.equals("out")) {
gui.magscale /= 2.0;
gui.auto_rescale = false;
gui.status("auto_rescale="+gui.auto_rescale);
}
else
if (command.equals("Out")) {
gui.magscale /= 6.0;
gui.auto_rescale = false;
gui.status("auto_rescale="+gui.auto_rescale);
}
else
if (command.equals("up")) {
gui.Window.YOffset += 150/gui.magscale;
gui.auto_rescale = false;
gui.status("auto_rescale="+gui.auto_rescale);
}
else
if (command.equals("Up")) {
gui.Window.YOffset += 450/gui.magscale;
gui.auto_rescale = false;
gui.status("auto_rescale="+gui.auto_rescale);
}
else
if (command.equals("do")) {
gui.Window.YOffset -= 150/gui.magscale;
gui.auto_rescale = false;
gui.status("auto_rescale="+gui.auto_rescale);
}
else
```

05/23/00  
00:16:51

# Confidential and Proprietary Property of Rocky Nevin

## Input.java

```
if (command.equals("Do")) {
    gui.WindowYOffset -= 450/gui.magscale;
    gui.auto_rescale = false;
    gui.status("auto_rescale = "+gui.auto_rescale);
}
else
if (command.equals("le")) {
    gui.WindowXOffset -= 150/gui.magscale;
    gui.auto_rescale = false;
    gui.status("auto_rescale = "+gui.auto_rescale);
}
else
if (command.equals("Le")) {
    gui.WindowXOffset -= 450/gui.magscale;
    gui.auto_rescale = false;
    gui.status("auto_rescale = "+gui.auto_rescale);
}
else
if (command.equals("n")) {
    gui.WindowXOffset += 150/gui.magscale;
    gui.auto_rescale = false;
    gui.status("auto_rescale = "+gui.auto_rescale);
}
else
if (command.equals("Ri")) {
    gui.WindowXOffset += 450/gui.magscale;
    gui.auto_rescale = false;
    gui.status("auto_rescale = "+gui.auto_rescale);
}
else
if (command.equals("s")) {
    gui.spread_factor *= 0.9;
}
else
if (command.equals("S")) {
    gui.spread_factor *= 1.1;
}
else
if (command.equals("q")) {
    gui.quit();
}
else
if (command.equals("m") || command.equals("more")) {
    gui.datasea.moreless("m");
}
else
```

```
if (command.equals("I") || command.equals("less")) {
    gui.datasea.moreless("I");
}
else
if (command.equals("r")) {
    gui.mode.obj.toggle.draw_text();
}
/*****
else
if (command.equals("setiny")) {
    gui.TinyNode = gui.lastNode;
    if (gui.TinyNode != null) {
        if (gui.TinyNode.TinyScale == 1.0)
            gui.TinyNode.TinyScale = 0.2;
        else
            gui.TinyNode.TinyScale = 1.0;
        GUI.P(0, "word_selector", "TinyNode "+gui.TinyNode.Name+" Scale
            gui.TinyNode.TinyScale);
    }
}
else
if (command.equals("makeiny")) {
    gui.TinyNode = gui.lastNode;
    if (gui.TinyNode != null) {
        if (gui.TinyNode.TinyScale == 1.0)
            gui.TinyNode.TinyScale = 0.2;
        else
            gui.TinyNode.TinyScale = 1.0;
        GUI.P(0, "word_selector", "TinyNode "+gui.TinyNode.Name+" Scale
            gui.TinyNode.TinyScale);
    }
}
else
if (command.equals("tiny")) {
    gui.TinyFlag = !gui.TinyFlag;
    GUI.P(0, "word_selector", "TinyFlag is "+gui.TinyFlag);
}
else
if (command.equals("details")) {
    gui.Details = !gui.Details;
    gui.status("Details = "+gui.Details);
}
}
*****/
```

05/23/00  
00:16:51

## Confidential and Proprietary Property of Rocky Nevin

Input.java

11

```
        else
            if (command.equals("update")) {
                gui.datasea.needdistUpdate = true;
            }
        else
            if (command.equals("parse")) {
                gui.parse = !gui.parse;
                GUI.P(0, "word_selector", "parse is "+gui.parse);
                gui.status("parse is "+gui.parse);
            }
        else
            if (command.equals("debug")) {
                if (gui.lastNode != null) {
                    gui.lastNode.Debug = !gui.lastNode.Debug;
                    GUI.P(0, "word_selector", "lastNode("+gui.lastNode.Name+", Debug is "+gui.lastNode.Debug);
                }
            }
        else
            if (command.equals("diag")) {
                gui.diag.frame.setVisible(true);
            }
        else
            if (command.equals("noddiag")) {
                gui.diag.frame.setVisible(false);
            }
        else
            if (command.equals("u")) {
                gui.datasea.undo(1);
            }
        else
            if (command.equals("smp")) {
                gui.datasea.simplify(words, num_words);
            }
        else
            if (command.equals("rim")) { // m=minus
                gui.datasea.vote_branch(words, num_words, "-");
            }
        else
            if (command.equals("grow")) { // p=plus
                gui.datasea.vote_branch(words, num_words, "+");
            }
        else
            if (command.equals("yes")) { // p=plus
                gui.datasea.vote_branch(words, num_words, "+");
            }
    }

    else
        if (command.equals("no")) { // m=minus
            gui.datasea.vote_branch(words, num_words, "-");
        }
    else
        if (command.equals("boxes")) {
            gui.drawBoxes = !gui.drawBoxes;
            GUI.P(0, "word_selector", "drawBoxes is "+gui.drawBoxes);
        }
    else
        if (command.equals("4")) {
            gui.datasea.reset_mags(Node.MAX_MAG);
        }
    else
        if (command.equals("3")) {
            gui.datasea.reset_mags(Node.BIG_MAG);
        }
    else
        if (command.equals("2")) {
            gui.datasea.reset_mags(Node.MED_MAG);
        }
    else
        if (command.equals("1")) {
            gui.datasea.reset_mags(Node.BARELY_VISIBLE_MAG+2);
        }
    else
        if (command.equals("0")) {
            gui.datasea.reset_mags(Node.BARELY_INVISIBLE_MAG);
        }
    else
        if (command.equals("x")) {
            gui.coordinates.on = !gui.coordinates.on;
        }
    else
        if (command.equals("fla")) {
            gui.datasea.flatten();
        }
    else
        if (command.equals("sha")) {
            gui.datasea.sharpen();
        }
    else
        if (command.equals("norm")) {
            gui.datasea.normalize();
        }
    else
```

```

    if (command.equals("newframe")) {
        gui.extra_frame();
    }
    else
    if (command.equals("test")) {
        gui.datasea.test();
    }
    else
    if (command.equals("stopspread")) {
        if (gui.lastNode != null) {
            gui.lastNode.StopSpread = true;
            gui.P(0, "input", "StopSpread true for "+gui.lastNode.Name);
        }
    }
    else
    if (command.equals("isolate")) {
        gui.datasea.isolate();
    }
    else
    if (command.equals("clean")) {
        gui.datasea.clean();
    }
    else
    if (command.equals("like")) {
        gui.datasea.like(gui.lastNode);
    }
    else
    if (command.equals("inh")) {
        gui.datasea.inhibit(words, num_words);
    }
    else
    if (command.equals("inhd")) { // inh all distally to infinity
        gui.datasea.inhd();
    }
    else
    if (command.equals("drill")) {
        gui.datasea.drill(words, num_words);
    }
    else
    if (command.equals("set")) {
        gui.datasea.parse_set_cmd(words, num_words);
    }
    else
    if (command.equals("recent")) {
        gui.datasea.recent();
    }
}

    else
    if (command.equals("lm")) {
        gui.mode.obj.toggle_line_mode();
    }
    else
    if (command.equals(".")) {
        gui.datasea.repeat_priorCommand();
    }
    else
    if (command.equals("rm")) {
        gui.mode.obj.toggle_render_mode();
    }
    else
    if (command.equals("pm")) {
        gui.mode.obj.toggle_position_mode();
    }
    else
    if (command.equals("sm")) {
        gui.mode.obj.toggle_spread_mode();
    }
    else
    if (command.equals("auto")) {
        gui.datasea.auto_flatten();
    }
    else
    if (command.equals("pol")) {
        gui.checkPolarization = !gui.checkPolarization;
        GUI.P(0, "word_selector", "gui.checkPolarization = "+gui.checkPolarization);
    }
    else
    if (command.equals("clear")) {
        gui.datasea.init();
    }
    else
    if (command.equals("clearstatus")) {
        gui.clear_status();
    }
    else
    if (command.equals("clearmsg")) {
        gui.global_str[0] = "";
        gui.global_str.size = 0;
    }
    else
    if (command.equals("action")) {
        gui.action(gui.lastNode);
    }
}

```

05/23/00  
00:16:51

## Confidential and Proprietary Property of Rocky Nevin

### Input.java

13

```
else
    if (command.equals("get")) {
        if (null != gui.lastNode)
            GUI.P0."word_selector", "getting:"+gui.lastNode.Name);
        gui.datasea.pop.get_a_URL(gui.lastNode);
    }
    else
        if (command.equals("readfile")) {
            gui.datasea.pop.read_in_net_file(0);
        }
        else
            if (command.equals("readfile1")) {
                gui.datasea.pop.read_in_net_file(1);
            }
            else
                if (command.equals("readfile2")) {
                    gui.datasea.pop.read_in_net_file(2);
                }
                else
                    if (command.equals("getfile")) {
                        gui.datasea.pop.get_file();
                    }
                    else
                        if (command.equals("tallis")) {
                            gui.datasea.pop.pull_in_URLs(String)"www.Tallis.com/", "TOC.html", 1, (Node)null);
                        }
                        else
                            if (command.equals("newrest")) {
                                gui.datasea.newrest();
                            }
                            else
                                if (command.equals("popprev")) {
                                    gui.datasea.pop.popprev();
                                }
                                else
                                    if (command.equals("poppegg")) {
                                        gui.datasea.pop.poppegg();
                                    }
                                    else
                                        if (command.equals("popiat")) {
                                            gui.datasea.pop.pull_in_URLs(String)"file:/usr/people/rocky/CIM/web/TallisB
ack/CIM.html", "", 1, (Node)null);
                                        }
                                        else
                                            if (command.equals("doocIM")) {

ex.html", 1, (Node)null);
        gui.datasea.pop.pull_in_URLs(String)"file:/usr/people/rocky/CIM/web/", "ind
        }
        else
            if (command.equals("popialis")) {
                gui.datasea.pop.pull_in_URLs(String)"http://www.Tallis.com/", "Profile.html"
                , 1, (Node)null);
            }
            else
                if (command.equals("popubc")) {
                    gui.datasea.pop.pull_in_URLs(String)"http://www.berkeley.edu/", "index.html
                    ", 1, (Node)null);
                }
                else
                    if (command.equals("showdist")) {
                        gui.datasea.showdist(words, num, words);
                    }
                    else
                        if (command.equals("run1")) {
                            gui.run1();
                        }
                        // populate POPULATE
                        else
                            if (command.equals("popframe")) {
                                gui.datasea.pop.popframe();
                                gui.datasea.needdistUpdate = true;
                            }
                            else
                                if (command.equals("popBB")) {
                                    gui.datasea.pop.popBB();
                                    gui.datasea.needdistUpdate = true;
                                }
                                else
                                    if (command.equals("popfiles")) {
                                        gui.datasea.pop.pull_in_URLs(String)null, (String)null, 1, (Node)null);
                                        gui.datasea.needdistUpdate = true;
                                    }
                                    else
                                        if (command.equals("pop1")) {
                                            gui.datasea.pop.pop.initial();
                                            gui.datasea.pop.popd();
                                            gui.datasea.pop.pops();
                                            gui.datasea.pop.popm();
                                            gui.datasea.pop.popff();
```

## Input.java

```
gui.datasea.pop.popn();
gui.datasea.pop.popx();
//gui.datasea.pop.pop.properties();
gui.datasea.needitUpdate = true;
}
else
if (command.equals("pop2")) {
    gui.datasea.pop.popEcon();
    gui.datasea.pop.popDM();
    gui.datasea.pop.popSemi();
    gui.datasea.pop.popURLs();
    gui.datasea.pop.popfab();
    gui.datasea.pop.popmap();
    gui.datasea.needitUpdate = true;
}
else
if (command.equals("popgroc")) {
    gui.datasea.pop.popgroc();
    gui.datasea.needitUpdate = true;
}
else
if (command.equals("popentl")) {
    gui.datasea.pop.popentl();
    gui.datasea.needitUpdate = true;
}
else
if (command.equals("popapps")) {
    gui.datasea.pop.popapps();
    gui.datasea.needitUpdate = true;
}
}
else
if (command.equals("popportal")) {
    gui.datasea.pop.popportal();
}
else
if (command.equals("pop")) {
    gui.datasea.pop.pop();
    gui.datasea.needitUpdate = true;
}
else
if (command.equals("poppp")) {
    gui.datasea.pop.poppp();
    gui.datasea.needitUpdate = true;
}
else
if (command.equals("popnet")) {
    gui.datasea.pop.popnet();
    gui.datasea.needitUpdate = true;
}
else
if (command.equals("popw")) {
    gui.datasea.pop.popweb();
    gui.datasea.needitUpdate = true;
}
else
if (command.equals("popinitial")) {
    gui.datasea.pop.pop.initial();
    gui.datasea.needitUpdate = true;
}
}
if (command.equals("popin")) {
    gui.datasea.pop.popin();
}
}
else
if (command.equals("popsynonyms")) {
    gui.datasea.pop.pop.synonyms();
    gui.datasea.needitUpdate = true;
}
else
if (command.equals("popEcon")) {
    gui.datasea.pop.popEcon();
    gui.datasea.needitUpdate = true;
}
}
else
if (command.equals("popDM")) {
    gui.datasea.pop.popDM();
    gui.datasea.needitUpdate = true;
}
}
else
if (command.equals("popSemi")) {
    gui.datasea.pop.popSemi();
    gui.datasea.needitUpdate = true;
}
}
else
if (command.equals("popURLs")) {
    gui.datasea.pop.popURLs();
    gui.datasea.needitUpdate = true;
}
}
else
if (command.equals("popi")) {
    gui.datasea.pop.popi();
    gui.datasea.needitUpdate = true;
}
}
```

```
        else
        if (command.equals("popTL")) {
            gui.datasource.pop.popTL();
            gui.datasource.neeDistUpdate = true;
        }
        else
        if (command.equals("popd")) {
            gui.datasource.pop.popd();
            gui.datasource.neeDistUpdate = true;
        }
        else
        if (command.equals("demo1")) {
            gui.demo1();
        }
        else
        if (command.equals("popm")) {
            gui.datasource.pop.popm();
            gui.datasource.neeDistUpdate = true;
        }
        else
        if (command.equals("demo2")) {
            gui.demo2();
        }
        else
        if (command.equals("rem")) {
            gui.datasource.reminder();
        }
        else
        if (command.equals("for")) {
            gui.datasource.forget();
        }
        else
        if (command.equals("peg")) {
            gui.datasource.peg();
        }
        else
        if (command.equals("pops")) {
            gui.datasource.pop.pops();
        }
        else
        if (command.equals("popf")) {
            gui.datasource.pop.popf();
        }
        else
        if (command.equals("demo3")) {
            gui.demo3();
        }
    }
}

    }
    else
    if (command.equals("popx")) {
        gui.datasource.pop.popx();
        gui.datasource.neeDistUpdate = true;
    }
    else
    if (command.equals("popn")) {
        gui.datasource.pop.popn();
        gui.datasource.neeDistUpdate = true;
    }
    else
    if (command.equals("popmap")) {
        gui.datasource.pop.popmap();
        gui.datasource.neeDistUpdate = true;
    }
    else
    if (command.equals("popfab")) {
        gui.datasource.pop.popfab();
        gui.datasource.neeDistUpdate = true;
    }
    else
    if (command.equals("popcal")) {
        gui.datasource.pop.popcalendar();
        gui.datasource.neeDistUpdate = true;
    }
    else
    if (command.equals("popc")) {
        gui.datasource.pop.popc();
        gui.datasource.neeDistUpdate = true;
    }
    else
    if (command.equals("why")) {
        gui.datasource.word.trace(words, num_words);
    }
    else
    if (command.equals("trace")) {
        gui.datasource.word.trace(words, num_words);
    }
    else
    if (command.equals("whats")) {
        gui.datasource.whats(words, num_words);
    }
    else
    understood = false;
    if (!understood)
```

05/23/00  
00:16:51

## Confidential and Proprietary Property of Rocky Nevin

### Input.java

16

```
d.");
    GUI.WARNING(0,"word_selector","Command '"+words[0]+' not understood.");

    //gui.datasource.needitUpdate = true;

    return(rel_node);
} // end word_selector

/**
 ** method word_mode.command set the spread mode
 */
public void word_mode.command (String[] words, int num_words) {
    if (num_words == 0) {
        GUI.ERROR(0,"word_mode.command","Zero words given.");
        return;
    }
    GUI.P(0,"word_mode.command","Got command '"+words[0]+'");

    if (words[0].equals("sr"))
        gui_mode_obj.set_spread_mode("radial");
    else if (words[0].equals("sd"))
        gui_mode_obj.set_spread_mode("distal");
    else if (words[0].equals("sp"))
        gui_mode_obj.set_spread_mode("proximal");

    //
    /*****
    else if (words[0].equals("lr"))
        gui_mode_obj.toggle_lines_mode("radial");
    else if (words[0].equals("ld"))
        gui_mode_obj.toggle_lines_mode("distal");
    else if (words[0].equals("lp"))
        gui_mode_obj.toggle_lines_mode("proximal");
    *****/

    //
    else if (words[0].equals("snap"))
        gui_mode_obj.set_render_mode("VR");
    else if (words[0].equals("nosnap"))
        gui_mode_obj.set_render_mode("relations");
    //
    else if (words[0].equals("pr"))
        gui_mode_obj.set_position_mode("relations");
    else if (words[0].equals("pl"))
        gui_mode_obj.set_position_mode("levels");
}

else
    GUI.ERROR(0,"word_mode.command","Unknown command '"+words[0]+'");

} // end word_mode.command

/**
 ** method words_input
 */
public Node words_input (String[] passed_words, String input_string, int num_words) {
    Node note_node=null;

    // put the string into a note_node without the first word 'input'
    String s = "";
    if (num_words > 1)
        s = passed_words[1];
    for (int i=2;i<num_words;i++)
        s = s + " "+passed_words[i];

    // check the CN 'Notes'
    if (gui.datasource.Notes == null) {
        gui.datasource.Notes = gui.datasource.pop_create_node("Notes", "AN");
    // HERE
        //gui.datasource.Notes.isCN = true;
    }

    Node triplet_node = search_for_possessive(passed_words, note_node, note_node);
    // ==>
    if (triplet_node == null) {
        note_node = gui.datasource.pop_create_node(s, "DN");
        gui.datasource.Notes.link(note_node); // link the whole text node to 'Notes'
        check_times(passed_words, gui.datasource.Notes.Name, note_node);
        String[] Str2 = discard_words(passed_words);
        parse_and_link(Str2, note_node, note_node);
    }

    return(note_node);
} // end words_input
```

```

/**
 ** check_times look for time words and link given node to actual time referred to
 **
 */
public void check_times (String[] passed_words, String CNode.name, Node node) {

    int num_words = passed_words.length;

    for (int i=0; i< num_words; i++) {
        // Search for time words
        if (passed_words[i].equalsIgnoreCase("yesterday")) {
            gui.il.create_TS(node, CNode.name, "Mar 11 2000");
        }
        //System.out.println("check_times: linking 'Mar 11 2000' (yesterday) to <"+node.Name+">");
        else if (passed_words[i].equalsIgnoreCase("today")) {
            gui.il.create_TS(node, CNode.name, "Mar 12 2000");
        }
        //System.out.println("check_times: linking 'Mar 12 2000' (today) to <"+node.Name+">");
        else if (passed_words[i].equalsIgnoreCase("tomorrow")) {
            gui.il.create_TS(node, CNode.name, "Mar 13 2000");
        }
        //System.out.println("check_times: linking 'Mar 13 2000' (tomorrow) to <"+node.Name+">");
        else if (passed_words[i].equalsIgnoreCase("1999")) {
            gui.il.create_TS(node, CNode.name, array.to_string(passed_words));
        }
        //System.out.println("check_times: checking times for <"+node.Name+">");
        else if (passed_words[i].equalsIgnoreCase("2000")) {
            gui.il.create_TS(node, CNode.name, array.to_string(passed_words));
        }
        //System.out.println("check_times: checking times for <"+node.Name+">");
    }
    // end for i
} // end check_times

/**
 ** search_for_possessive
 **
 */
public Node search_for_possessive (String[] passed_words, Node central_node, Node CNode) {
    int i, index, num_words=0;
    Node child;
    Node ret_node=null;

    num_words = passed_words.length;

    // Search for possessive "s" sign, create AN-DN combination for the possessive
    for (i=0; i<num_words; i++) {
        if (0 < (index = passed_words[i].indexOf("s"))) {
            //ret_val = true;
            GUI.P(1, "search_for_possessive", num_words="+num_words+", i="+i);
            GUI.P(1, "search_for_possessive", passed_words[i]="+passed_words[i]");
            if (num_words < (i+1+3)) {
                GUI.WARNING(0, "search_for_possessive", "Too few words");
            }
            else {
                String word1 = passed_words[i].substring(0,index); // BOB's hair is red
                String word2 = passed_words[i+1]; // bob's HAIR is red
                String word3 = passed_words[i+3]; // bob's hair is RED
                String desc = "is";
                if (passed_words[i+2].equalsIgnoreCase("is")) {
                    if (num_words>=(i+1+4)) // see if there are enough words to check
                        if (passed_words[i+3].equalsIgnoreCase("a")) {
                            desc = "is a"; // change description
                            word3 = passed_words[i+4]; // correct word3
                        }
                }
                // w1 <--> w3 <--> w2
                GUI.P(1, "search_for_possessive",
                    "POSSESSIVE: word1="+word1
                    +" <--> word2="+word2
                    +" <--> desc="+desc
                    +" <--> word3="+word3);
                ret_node = gui.datasea.pop.triple(word1, word3, word2);
                //*****
                context_node.link(gui.datasea.c(word1));
                context_node.link(gui.datasea.c(word2));
                context_node.link(gui.datasea.c(word3));
            }
            Node w2_node=null;
            if (null != (w2_node=gui.datasea.find_node.named(word2, "AN")))
                w2_node.set_Desc(gui.datasea.find_node.named(word3, "DN"), desc);
            //*****
        }
    }
    return(ret_node);
} // end search_for_possessive

```

05/23/00  
00:16:51

## Confidential and Proprietary Property of Rocky Nevin

Input.java

18

```
/**
 ** parse_and_link
 **
 */
public void parse_and_link (String[] passed_words, Node central_node, Node CNode) {
    int i, num_words;
    Node child, word_node=null;

    num_words = passed_words.length;

    // LINK INDIVIDUAL WORDS TO PASSED CNode
    for (i=0; i< num_words; i++) {
        if (passed_words.equals("")) // blank words have been 'discarded' previously
            ;
        else {
            //System.out.println("parse_and_link: linking <"+passed_words[i]+> to <"+central_node.Name+">");
            if (null == (word_node=gui_datasea.find_node_named(passed_words[i])))
                word_node = gui_datasea.pop_create_node(passed_words[i], "DN");
            central_node.link(word_node);
            //central_node.link(word_node, CNode); // this makes the Note node a CNode
            //gui_datasea.addCNodeBetweenNodes(central_node, word_node, CNode);
        }
        // end for i
    } // end parse_and_link

    /**
     ** is_useless_word
     **
     */
    public boolean is_useless_word (String word) {
        int i, size;
        Node child;

        if (word == null) {
            GUI.WARNING(0, "is_useless_word", "null word.");
            return(true);
        }

        if (0 <= gui_datasea.useless_words.indexOf(lower_case_word) ) {
            //System.out.println("Useless: "+lower_case_word);
            return (true);
        }
        else
            return(false);
    } // end is_useless_word

    /**
     ** discard_words clobber useless words
     **
     */
    public String[] discard_words (String[] passed_words) {
        int i, j=0, num_words;
        String[] ret_words;

        num_words = passed_words.length;
        String[] good_words = new String[num_words];

        for (i=0; i< num_words; i++) {
            if (is_useless_word(passed_words[i])) { // Search for useless words
            }
            else
                {
                    good_words[j++] = passed_words[i];
                    //System.out.println(passed_words[i]+");
                }
        } // end for i

        // THIS IS STUPID, BUT I WANT TO RETURN AN ARRAY OF ONLY GOOD WORDS
        ret_words = new String[j];
        for (i=0; i<j; i++)
            ret_words[i] = good_words[i];
        return(ret_words);
    } // end discard_words

    /**
     ** discard_words String input and output version
     **
     */
    public String discard_words (String s) {
        int i;
```

String lower\_case\_word = word.toLowerCase();

05/23/00  
00:16:51

## Confidential and Proprietary Property of Rocky Nevin

Input.java

19

```
String[] words = string_to_array(s);
```

```
// CALL THE STRING[] VERSION OF OURSELF
```

```
String[] ret_words = discard_words(words);
```

```
// REFORMAT BACK TO ORIGINAL FORM
```

```
String ret_string = array_to_string(ret_words); // when do we have to say 'new'?
```

```
return(ret_string);
```

```
} // end string version of discard_words
```

```
/**
```

```
 ** array_to_string
```

```
 **
```

```
 */
```

```
public String array_to_string (String[] str_array) {
```

```
int i;
```

```
// PUT BACK INTO STRING
```

```
String ret_string=""; // when do we have to say 'new'?
```

```
for (i=0; i<str_array.length; i++)
```

```
    ret_string = ret_string+str_array[i]+" ";
```

```
return(ret_string);
```

```
} // end array_to_string
```

```
/**
```

```
 ** array_to_string
```

```
 **
```

```
 */
```

```
public String[] string_to_array (String s) {
```

```
int i;
```

```
// TOKENIZE
```

```
StringTokenizer st = new StringTokenizer(s, ".,?<>\\\"'\\n");
```

```
int num_words = st.countTokens();
```

```
String[] words = new String[num_words];
```

```
for (i=0; i<num_words; i++)
```

```
    words[i] = st.nextToken();
```

```
return(words);
```

```
} // end array_to_string
```

```
/**
```

```
 ** parse data
```

```
 **
```

```
 */
```

```
public void parse_data (Node node) {
```

```
int i, j, size, num_words=0, index=0, counter=0;
```

```
Node child;
```

```
String words[];
```

```
String line;
```

```
if (!gui.parse) {
```

```
    return;
```

```
}
```

```
if (node==null) {
```

```
    return;
```

```
}
```

```
if (node.Data==null) {
```

```
    return;
```

```
}
```

```
if (node.Data[0]==null) {
```

```
    return;
```

```
}
```

```
for (i=0; i<Node.MAX_TEXT_DATA_LINES; i++) {
```

```
    line = node.Data[i];
```

```
    if (line == null)
```

```
        break;
```

```
    line = gui.eliminate_html(line);
```

```
    words = string_to_array(discard_words(line));
```

```
    num_words = words.length;
```

```
    for (j = 0; j < num_words; j++) {
```

```
        child = gui.datasea.pop_create_node(words[j], "DN");
```

```
        //System.out.println(node.Name+" "+words[j]);
```

```
        node.link(child, "");
```

```
    }
```

```
}
```

```
return;
```

```
} // end parse_data
```

```
} // end Input
```

```
// This is Populate.java      by Rocky Nevlin

import java.util.*;
import java.io.*;
import java.awt.*;
```

## // POPULATING and CREATING NODES

```

public class Populate extends Object {

    DataSea dataset;

    static Node lastBnode; // a bad way, but, nonetheless, used by make_BB()
    static Node AllURLs = null;
    static Node Cluster_Files;
    static Node Cluster_Web;
    static Node URLs;
    static Node Cluster_Mail;
    static Node Cluster_Directory;
    static Node Cluster_Notes;
    static Node Cluster_Em;
    static Node Timeline;
    static Node yesterday_node, today_node, tomorrow_node;
    static Node last_week_node, this_week_node, next_week_node;
    static int MAX_FILES_PER_DIRECTORY=15;
    static int MAX_DIRECTORY_DEPTH=4;
    static int global_counter=0;
    static int counter=0;
    static int calls_to_pull=0;
    //static int file_counter=0;
    GetURLInfo url;
}

```

```
public Populate (DataSea passed_datasea_obj) { // Constructor
    datasea = passed_datasea_obj;
    this.init();
} // end Populate constructor
```

```
public void init () {
    public void populate_begin () {
        GUI.sleep(100);
        System.err.println("Populate.populate_begin() begun...");
        //GUI.sleep(100);
        pop.initial();
        read in net file(0);
    }
}
```

```
//GUI.sleep(100);
//GUI.sleep(100);
//
//    popportal();
//GUI.sleep(100);
//
//    popd();
//GUI.sleep(100);
//
//    popgeneology();
//GUI.sleep(100);
//
//    popp();
//GUI.sleep(100);
//
//    popTL();
//GUI.sleep(100);
//
//    popcalendar();
//GUI.sleep(100);
//
//    popn();
//
//end populate_begin
//end populate_begin

System.err.println("Populate-populate_begin done.");
```

```

/**
 ** get_file
 **
 */

public void get_file () {
    GetURLInfo uri;
    byte byteArray[]=new byte[300];
    int i, index;
    String w1=null, w2=null;
    int max_lines = 700;
    FileInputStream file;

    try { file = new FileInputStream("/usr/people/rocky/x"); }
    catch (java.io.FileNotFoundException e)

    {
        GUI.WARNING(0,"get_file" , ->"+e.toString()+"<-");
        return;
    }

    try { file.read(byteArray); }
    catch (java.io.IOException e)

    {
        GUI.WARNING(0,"get_file" , ->"+e.toString()+"<-");
        return;
    }
}

system.out.println("byte array from /usr/people/rocky/x:");

```

05/23/00  
00:16:59

## Confidential and Proprietary Property of Rocky Nevin Populate.java

2

```
System.out.println(bytearray);

} //get_file

/**
 ** read_in_net_file
 **
 */
public void read_in_net_file(int which) {
    GetURLInfo url;
    String str_array[]=new String[0];
    int i, index;
    String w1=null, w2=null, name, value;
    int max_lines = 700;
    String FileName = "/README";

    if (which == 1)
        FileName = "/README.1";
    else
        if (which == 2)
            FileName = "/README.2";

    url = new GetURLInfo();
    try { str_array = url.get_URL("file:"+GUI.GlobalUserDir+FileName, max_lines); }
    catch (java.net.MalformedURLException e)
    {
        GUI.WARNING(0,"read_in_net_file" -> "+e.toString()+"<-");
        return;
    }
    catch (IOException e)
    {
        GUI.WARNING(0,"read_in_net_file" -> "+e.toString()+"<-");
        return;
    }

    if (str_array == null) {
        GUI.WARNING(0,"read_in_net_file", "str_array is null.");
        return;
    }

    DataSea.currentCNode = create_node("ThesCN", "CN"); // Turn on auto-CNNode links
    for (i=0; i<max_lines; i++) {
        if (str_array[i] == null)
            break;
    }
}

if (str_array[i].length() > 0)
    if (!str_array[i].substring(0,1).equals("#")) {
        index = str_array[i].indexOf(",");
        if (index <= 0)
            break;
        w1 = str_array[i].substring(0,index);
        w2 = str_array[i].substring(index+2);

        //meta subject:subject value
        if (0<=w1.indexOf("meta")) {
            index = w1.indexOf(":");
            name = w1.substring(5, index);
            value = w1.substring(index+1);
            METAPair(name, value, w2);
        }
        else
            if (0<=w1.indexOf("http"))
                if (0<=w2.indexOf("http"))
                    URLtoURLpair(w1, w2);
                else
                    URLtoANPair(w1, w2);
            else
                ANANPair(w1, w2);

        } // if not '#'
    } // for i
    System.err.println("Total lines: "+i);

    DataSea.currentCNode = null; // Turn off auto-CNNode links
    return;
} // end get_file.input

/**
 ** pull_in_URLs
 **
 */
public void pull_in_URLs (String base, String name, int current_depth, Node parent) {
    String str_array[]=new String[0];
```

05/23/00  
00:16:59

## Confidential and Proprietary Property of Rocky Nevin

### Populate.java

3

```
String url_name="UNKNOWN";
String domain_name=null;
String line;
int i;
int index=0, end=0, end_space, end_bracket;
int MAX_CHILDREN = 7-current_depth;
int MAX_DEPTH = 4; // set conditionally below on the OS
int child_count=0;
Node node=null, cn_node=null;

if (current_depth > MAX_DEPTH) {
    return;
}
if (current_depth == 1)
    calls_to_pull = 0;

if (parent == null) // starting, so link first to a reliable node
if (null == (parent = datasea.find_node_named("URLs"))) {
    (parent = create_node("URLs", "DN")).link(datasea.Root);
}

if (AllURLs == null)
    AllURLs = create_node("AllURLs", "DN"); // link all to this one

if ((System.getProperty("os.name").equalsIgnoreCase("trix")) {
    MAX_DEPTH = 3;
    if (base == null)
        base = "file:./././CIM/web/";
    }
else {
    MAX_DEPTH = 6; // Using cable-modem on Wintel
    if (base == null)
        base = "file:./";
    }
    if (name == null)
        name = "", // just get the files

    if (calls_to_pull++ > 30)
    {
        datasea.gui.P(0, "pull_in_URLs", "calls_to_pull = "+calls_to_pull);
        return;
    }
    url_name = base+name;

//System.err.println("1) STARTING parent ->"+parent.Name+"<-");
//System.err.println("1)      base is ->"+base+"<-");
//System.err.println("1)      name is ->"+name+"<-");
    if ((0<url_name.toLowerCase().indexOf("http") && !datasea.gui.NetOK) {
        datasea.gui.P(0, "pull_in_URLs",
            "Sensed http, and GUI.NetOK is false, so aborting data retrieval for "+url_name);
        return;
    }
    i = 1 + url_name.lastIndexOf("/"); // recalculate the base and the file name
    if (i>0) {
        base = url_name.substring(0,i);
        name = url_name.substring(i);
        //System.err.println(parent.Name+" "+url_name);
        //System.err.println("      base is ->"+base+"<-, name is ->"+name+"<-");
        //System.err.println("      name is ->"+name+"<-");
    }
    url = new GetURLInfo();
    try { str_array = url.get_URL(url_name); }
    catch (java.net.MalformedURLException e)
    {
        //GUI.WARNING(0, "pull_in_URLs", "->"+e.toString()+"<-");
        return;
    }
    catch (IOException e)
    {
        //GUI.WARNING(0, "pull_in_URLs", "->"+e.toString()+"<-");
        return;
    }
    if (str_array == null) {
        //GUI.WARNING(0, "pull_in_URLs", "str_array is null.");
        return;
    }
    // Create and link a node for this URL, if it doesn't exist already
    if (null == (node = datasea.find_node_named(url_name))) {
        node = create_node(url_name, "URL");
    }
    parent.link(node);
    //GUI.P(0, "pull_in_URLs", "linking AllURLs to "+node.Name);
    AllURLs.link(node); // link all to this one
}
```

05/23/00  
00:16:59

## Confidential and Proprietary Property of Rocky Nevin

### Populate.java

4

```
// figure out the domain name, use as a CN ...
//
if (null != (domain_name=dataset.gui.find_domain_name(url_name))) {
    if (null == (cn_node = dataset.find_node_named(domain_name))) {
        cn_node = create_node(domain_name, "CN");
    }
    cn_node.link(node);
    //AllURLs.link(cn_node);
    //GUI.P(0, "pull_in_URLS", "linking CN "+cn_node.Name);
}
else
    GUI.WARNING(0, "pull_in_URLS", "Failed to find good domain: domain_name is "+do
main_name);

node.isSelected = true;
dataset.gui.show_node_once(node);

for (i=0; i<Node.MAX_TEXT_DATA_LINES; i++) {
    if (str_array[i] == null)
        break;
    node.Data[i]=str_array[i];
    //System.out.println("Adding to "+node.Name+": "+str_array[i]);
}
GUI.input.parse_data(node);

if (child_count > MAX_CHILDREN) {
    System.err.print(".");
    node.isSelected = false;
    return;
}

// Now, look for URLs to call
String new_name="";
++current_depth;
if (current_depth > MAX_DEPTH) {
    node.isSelected = false;
    return;
}

for (i = 0; i < Node.MAX_TEXT_DATA_LINES; i++) {
    line = str_array[i];
    if (line == null) break;
    new_name = dataset.gui.find_URL_name(line);
    if (new_name != null) {
        if (0<new_name.toLowerCase().indexOf("http")) {
            null in URLs("", new_name, current_depth, node);
        }
        else {
            pull_in_URLs(base, new_name, current_depth, node);
        }
    }
    node.isSelected = false;
    dataset.gui.update();
} // end pull_in_URLs

}

/**
 ** pull_in_URLs no argument version
 **
 */
public void pull_in_URLs () {
    System.err.println("===== pull_in_URLs ... Begun =====");
    pull_in_URLs((String)null, (String)null, 1, (Node)null);
    System.err.println("===== pull_in_URLs Done =====");
    System.err.println("size of node_vec is "+dataset.node_vec.size());
} // end no arg version of pull_in_URLs

/**
 ** popMyFamily
 **
 */
public void popMyFamily () { // "show geneology"
    Node grape_node, cereal_node, nuts_node, wheat_node;

    // ANs: grape, cereal, nuts, red, green, wine, wheat, zinfandel, crunchy
    // DNS: Grape Nuts, Wheat Germ, Shredded Wheat, Acacia Zinfandel, Green Grapes, Red Grapes,
    //
    //
    Node geneologyCN = create_node("geneology", "AN");

    triplet("Rocky", "Jane Elinore GEIGER", "Mother", geneologyCN, "polarized");
    triplet("Rocky", "Harry Wardlaw Jr. NEVIN", "Father", geneologyCN, "polarized");
    triplet("Jane Elinore GIEGER", "Elinore Lucille LINGARD", "Mother", geneologyCN, "polarized");
    triplet("Jane Elinore GIEGER", "Charles Towne GEIGER", "Father", geneologyCN, "polarized");
}
```

```
triplet("Harry Wardlaw Jr. NEVIN", "Hazel Miller HAWKINS", "Mother", geneologyCN, "polarized");
triplet("Harry Wardlaw Jr. NEVIN", "Harry Wardlaw Sr. NEVIN", "Father", geneologyCN, "polarized");
triplet("Elinore Lucille LINGARD", "Eliza Jane BAKER", "Mother", geneologyCN, "polarized");
triplet("Elinore Lucille LINGARD", "Amos Lister LINGARD", "Father", geneologyCN, "polarized");

triplet("Charles Towne GEIGER", "Grace TIDRICK", "Mother", geneologyCN, "polarized");
triplet("Charles Towne GEIGER", "Eugene Warfel GEIGER", "Father", geneologyCN, "polarized");

triplet("Harry Wardlaw Sr. NEVIN", "Lula Wardlaw", "Mother", geneologyCN, "polarized");
triplet("Harry Wardlaw Sr. NEVIN", "Michael John NEVIN", "Father", geneologyCN, "polarized");

triplet("Hazel Miller HAWKINS", "Janetta A CLOUGH", "Mother", geneologyCN, "polarized");
triplet("Hazel Miller HAWKINS", "George Wilcon HAWKINS", "Father", geneologyCN, "polarized");

Node DirCNnode = create_node("DirCN", "CN");
triplet("Grace TIDRICK", "734 S.Main", "address", DirCNnode, "polarized");

return;
} // end popMyFamily

/**
 ** popgeneology
 **
 */
public void popgeneology () { // "show geneology"
Node grape_node, cereal_node, nuts_node, wheat_node;

// ANs: grape, cereal, nuts, red, green, wine, wheat, zinfandel, crunchy
// DNS: Grape Nuts, Wheat Germ, Shredded Wheat, Acacia Zinfandel, Green Grapes, Red Grapes,
//
//
Node geneologyCN = create_node("geneology", "AN");
Node employmentCN = create_node("employment", "AN");
employmentCN.isCN = true;
geneologyCN.isCN = true;
Node peopleAN = create_node("people", "AN");
DataSea.Root.link(geneologyCN);
DataSea.Root.link(employmentCN);

peopleAN.link(employmentCN, "polarized");
peopleAN.link(geneologyCN, "polarized");

triplet("Bob", "BobsDad", "Father", geneologyCN, "polarized");
triplet("Bob", "BobsSon", "Son", geneologyCN, "polarized");
triplet("BobsSon", "BobsGrandSon", "Son", geneologyCN, "polarized");

Node ann_node = create_node("Ann", "DN"); //
Node ted_node = create_node("Ted", "DN"); //
triplet("Bob", "Ann", "Wife", geneologyCN, "polarized");
triplet("Ann", "Ted", "Father", geneologyCN, "polarized");
triplet("Ted", "IBM", "employment", employmentCN, "polarized");

popMyFamily();

return;
} // end popgeneology

/**
 ** popgroc
 **
 */
public void popgroc () { // "show grocery"
Node grape_node, cereal_node, nuts_node, wheat_node;

// ANs: grape, cereal, nuts, red, green, wine, wheat, zinfandel, crunchy
// DNS: Grape Nuts, Wheat Germ, Shredded Wheat, Acacia Zinfandel, Green Grapes, Red Grapes,
//
//
Node grocery = create_node("grocery", "DN");
DataSea.Root.link(grocery);
grape_node = create_node("grape", "AN"); //
grocery.link(grape_node);
grape_node.link(datasea.cl("Grape Nuts", "DN"));
grape_node.link(datasea.cl("Red Grape", "DN"));
grape_node.link(datasea.cl("Green Grape", "DN"));

wheat_node = create_node("wheat", "AN"); //
grocery.link(wheat_node);
wheat_node.link(datasea.cl("Wheat Germ", "DN"));
wheat_node.link(datasea.cl("White Flour", "DN"));
wheat_node.link(datasea.cl("Shredded Wheat", "DN"));

cereal_node = create_node("cereal", "AN"); //
```

05/23/00  
00:16:59

## Confidential and Proprietary Property of Rocky Nevin

### Populate.java

6

```
grocery.link(cereal.node);
cereal.node.link(datasca.cl("Grape Nuts", "DN"));
cereal.node.link(datasca.cl("Shredded Wheat", "DN"));
cereal.node.link(datasca.cl("Corn Flakes", "DN"));
```

```
GUI.P(0, "popgroc", "show grocery");
return;
} // end popgroc
```

```
/**
 ** popcntl
 **
 */
public void popcntl () { // "show cntl"
Node file.node, edit.node, cntl.node, apps.node, pop.node;
```

```
// cntl - File - Open
//      - Save
//      - Export
//      - Import - File_1
//      - File_2
//      - File_3
//      - File_4
//      - Edit - Cut
//      - Copy
//      - Paste
//      - Select All
//      - Delete
```

```
cntl.node = create.node("Cntl", "AN"); // created also in popapps
apps.node = create.node("Apps", "AN");
apps.node.link(cntl.node);
DataSea.Root.link(cntl.node);
cntl.node.link(datasca.cl("File", "DN", "Open", "DN"));
cntl.node.link(datasca.cl("Edit", "DN", "Cut", "DN"));
file.node = datasca.find.node.named("File", "DN");
edit.node = datasca.find.node.named("Edit", "DN");
file.node.link(datasca.cl("Import", "DN", "File_1", "DN"));
(datasca.find.node.named("Import", "DN")).link(datasca.cl("File_2", "DN"));
(datasca.find.node.named("Import", "DN")).link(datasca.cl("File_3", "DN"));
(datasca.find.node.named("Import", "DN")).link(datasca.cl("File_4", "DN"));
file.node.link(datasca.cl("Export", "DN"));
file.node.link(datasca.cl("Save", "DN"));
file.node.link(datasca.cl("Save As", "DN"));
```

```
edit.node.link(datasca.cl("Copy", "DN"));
edit.node.link(datasca.cl("Paste", "DN"));
edit.node.link(datasca.cl("Select All", "DN"));
edit.node.link(datasca.cl("Delete", "DN"));
```

```
pop.node = create.node("Populate", "DN");
pop.node.link(create.node("pop", "DN"));
pop.node.link(create.node("popw", "DN"));
pop.node.link(create.node("popd", "DN"));
pop.node.link(create.node("popURLs", "DN"));
pop.node.link(create.node("popTL", "DN"));
pop.node.link(create.node("popapps", "DN"));
```

```
popapps();
```

```
GUI.P(0, "popcntl", "show cntl");
return;
} // end popcntl
```

```
/**
 ** popapps
 **
```

```
*/
public void popapps () { // "show apps"
Node mail.node, web.node, apps.node, cntl.node;
```

```
// Apps - Mail - Receive
//      - Send
//      - Cleanup
//      - Include File - Mail_1
//      - Mail_2
//      - Mail_3
//      - Mail_4
//      - Web - Open Browser
//      - Properties
//      - Review Bookmarks
//      - Home
//      - Blah
```

```
DataSea.currentCNode = create.node("AppsCN", "CN"); // Turn on auto-CNode links
```

```
cntl.node = create.node("Cntl", "AN"); // created also in popmenu
apps.node = create.node("Apps", "AN");
apps.node.link(cntl.node);
DataSea.Root.link(apps.node);
```

05/23/00  
00:16:59

## Confidential and Proprietary Property of Rocky Nevin Populate.java

7

```
apps.node.link(dataseta.cl("Mail", "DN", "Receive", "DN"));
apps.node.link(dataseta.cl("Web", "DN", "Open Browser", "DN"));
mail_node = create_node("Mail", "DN");
web_node = create_node("Web", "DN");
mail_node.link(dataseta.cl("Include File", "DN", "Mail.1", "DN"));
(create_node("Include File", "DN").link(dataseta.cl("Mail.2", "DN"));
(create_node("Include File", "DN").link(dataseta.cl("Mail.3", "DN"));
(create_node("Include File", "DN").link(dataseta.cl("Mail.4", "DN"));
mail_node.link(dataseta.cl("Cleanup", "DN"));
mail_node.link(dataseta.cl("Save", "DN"));
mail_node.link(dataseta.cl("Save As", "DN"));
web_node.link(dataseta.cl("Properties", "DN"));
web_node.link(dataseta.cl("Review Bookmarks", "DN"));
web_node.link(dataseta.cl("Home", "DN"));
web_node.link(dataseta.cl("Blah", "DN"));

GUI.P(0, "popapps", "show popapps");

DataSea.current(CNode = null; // Turn off auto-CNode links

return;
} // end popapps

/**
 ** properties
 **
 */
public void properties () {
    CNode prop_node;

    GUI.P(0, "properties", "Creating Property nodes ('Props')");

    prop_node = create_node("Props", "DN", "From System.getProperty() and Toolkit.getDefaultToolkit().getScreenSize()");
    DataSea.Root.link(prop_node);

    triple("Props", System.getProperty("java.version"), "JavaVersion");
    triple("Props", System.getProperty("java.vendor"), "JavaVendor");
    triple("Props", System.getProperty("java.class.version"), "JavaVersionURL");
    triple("Props", System.getProperty("java.class.version"), "JavaClassVersion");
    triple("Props", System.getProperty("os.name"), "OS_Name");
    triple("Props", System.getProperty("file.separator"), "FileSeparator");
    triple("Props", System.getProperty("user.name"), "UserName");
```

```
triple("Props", System.getProperty("user.home"), "UserHome");
triple("Props", System.getProperty("user.dir"), "UserDir");
triple("Props", "+Toolkit.getDefaultToolkit().getScreenSize()", "ScreenSize");

// GUI.GlobalUserDir = System.getProperty("user.dir");
// GUI.GlobalOSName = System.getProperty("os.name");
} // end properties

/**
 ** pop initial
 **
 */
public void pop_initial () { // "show root"
    Node n1, n11, n12, n2, n21, n22, n3;
    Node Cnt1;

    GUI.P(0, "pop_initial", "begun");

    if (DataSea.node_vec != null)
        DataSea.node_vec.removeAllElements();
    DataSea.node_vec = new Vector();

    GUI.selected_nodes_vec = new Vector(10);

    DataSea.Root = new Node("Root", "Layer", "", 0, 50, 10, 10);

    DataSea.Thesaurus_node = new Node("THES", "AN", "", 0, 0, 10, 10);
    Timeline = new Node("TL", "Event", "Timeline",
        -200, -100, 3, 400);
    //DataSea.Root.link(Timeline);
    //((DataSea.Root.linkTo(Timeline)).setLinksVRparms(Timeline);

    Cnt1 = new Node("Cnt1", "AN", "Control",
        -300, 50, 3, 3);
    Cluster_Files = new Node("Files", "AN", "Files",
        -200, 50, 3, 3);
    Cluster_Web = new Node("Web", "AN", "Web URLs",
        -100, 50, 3, 3);
    URLs = new Node("URLs", "CN", "URL CNode");
    Cluster_Mail = new Node("EMail", "AN", "E-mail",
        0, 0, 3, 3);
```

05/23/00  
00:16:59

## Confidential and Proprietary Property of Rocky Nevin

### Populate.java

8

```
Cluster.Directory = new Node("Dir", "AN", "Directory",
0, 0, 3, 3);
Cluster.Notes = new Node("Notes", "AN", "Free-form Notes",
0, 0, 3, 3);
Cluster.En = new Node("Encyclopedia", "AN", "Encyclopedia Online",
300, 50, 3, 3);

// Cluster.Mail.link(n1=create_node("Read-Mail", "AN"));
// n1.link(create_node("file:/usr/people/rocky/BackCode/Mail_DS/lt-1.html", "URL"));
// n1.link(create_node("file:/usr/people/rocky/BackCode/Mail_DS/lt-2.html", "URL"));
// n1.link(create_node("file:/usr/people/rocky/BackCode/Mail_DS/lt-3.html", "URL"));

Cluster.Web.link(create_node("http://www.metmuseum.org/htmlfile/faq/faq.html", "URL"));

DataSea.Root.StopsSpread = true; // caught by recursive routines, block spreading
URLs.StopsSpread = true;
Cluster.Web.StopsSpread = true;
Cluster.Mail.StopsSpread = true;
Cluster.Notes.StopsSpread = true;
Cluster.En.StopsSpread = true;
Cntl.StopsSpread = true;
Timeline.StopsSpread = true;

DataSea.Root.link(DataSea.Thesaurus_node);
DataSea.Root.link(Cluster.Files);
DataSea.Root.link(Cluster.Web);
DataSea.Root.link(Cluster.Mail);
DataSea.Root.link(Cluster.Notes);
DataSea.Root.link(Cluster.Directory);
DataSea.Root.link(Cluster.En);
DataSea.Root.link(Cntl);

DataSea.Root.setLinksVRparamsTo(Cluster.Files);
//DataSea.Root.getLinkTo(Cluster.Files).setLinksVRparams(Cluster.Files);
(DataSea.Root.getLinkTo(Cluster.Web)).setLinksVRparams(Cluster.Web);
(DataSea.Root.getLinkTo(Cluster.Mail)).setLinksVRparams(Cluster.Mail);
(DataSea.Root.getLinkTo(Cluster.Directory)).setLinksVRparams(Cluster.Directory);
(DataSea.Root.getLinkTo(Cluster.Notes)).setLinksVRparams(Cluster.Notes);
(DataSea.Root.getLinkTo(Cluster.En)).setLinksVRparams(Cluster.En);
/*****
Cntl.link(datasca.cl("Cmds", "AN", "", 0, 10));
Cntl.link(datasca.cl("Graphics", "AN", "", 20, 30));
Cntl.link(datasca.cl("Thresh", "AN", "", 40, 50));
Cntl.link(datasca.cl("demos", "AN", "", 60, 70));
Cntl.link(datasca.cl("Cmds", "AN", "", 80, 80));
*****/

*****/
n1 = new Node("name", "AN");
n11 = new Node("Rocky", "DN", "This is a test Node, named 'Rocky'");
n1.link(n11);
n12 = new Node("Bob", "DN");
n1.link(n12);

triple("CommitteeX", "Bob", "Member");

datasca.needitistUpdate = true;
GUI.P(0, "pop.initial", "done.");
} // end pop.initial

/**
** pop.synonyms
**
*/
public void pop.synonyms () { // "show piano"

    GUI.input.string.input("input piano:syn:klavier");
    GUI.input.string.input("input phone:syn:telephone");
    GUI.P(0, "pop.synonyms", "show piano");
} // end pop.synonyms

/**
** semi Populate Semiconductor fab
**
*/
public void popSemi () { // skip
    Node Semi, t_node;

    Semi = (DataSea.Root.link(datasca.cl("SemiConductor", "AN", "", 0, 0)).link(datasca.cl("SemiConductor Mfg", "AN", "", 0, 0)));
    Semi.link(datasca.cl("Semi Process", "AN", "", 0, 0));
    t_node = Semi.link(datasca.cl("www.semi.com", "DN", "URL", 0, 0));
    t_node = Semi.link(datasca.cl("URL-A", "DN", "URL", 0, 0));
    t_node = Semi.link(datasca.cl("URL-B", "DN", "URL", 0, 0));
    t_node = Semi.link(datasca.cl("URL-C", "DN", "URL", 0, 0));
    t_node = Semi.link(datasca.cl("PhD Thesis", "DN", "URL", 0, 0));
}
```

## Populate.java

```
        t_node.link(datasca.find_node.named("URL-A"));
        t_node.link(datasca.find_node.named("URL-B"));
        t_node.link(datasca.find_node.named("URL-C"));
        t_node.link(datasca.find_node.named("www.semi.com"));
        (datasca.find_node.named("URL-A")).link(datasca.find_node.named("Semi Process"));
        *****/

    } // end popSemi

    /**
     ** popDM Populate Data Mining
     **
     */
    public void popDM () { // skip
        Node DM, tnode;
        int i;

        GUI.P(0,"DM", "DataMining Run.");
        /*****
        // Master node
        DM = DataSea.Root.link(datasca.cl("DM", "AN", "DataMining",-20,20));

        // Gender node
        tnode = DM.link(datasca.cl("Gender", "AN", "Stats",-40,30));
        tnode.link(datasca.cl("Male", "DN", "Stats",-10,10));
        tnode.link(datasca.cl("Female", "DN", "Stats",10,10));

        // Dept node
        tnode = DM.link(datasca.cl("Dept", "AN", "Stats",-20,30));
        tnode.link(datasca.cl("Sports", "DN", "",-10,10));
        tnode.link(datasca.cl("Clothing", "DN", "",0,10));
        tnode.link(datasca.cl("Appliances", "DN", "",10,10));

        // Season node
        tnode = DM.link(datasca.cl("Season", "AN", "Stats",0,30));
        tnode.link(datasca.cl("Fall", "DN", "",-10,10));
        tnode.link(datasca.cl("Winter", "DN", "",10,10));
        tnode.link(datasca.cl("Spring", "DN", "",-10,10));
        tnode.link(datasca.cl("Summer", "DN", "",10,10));

        // Cost node
        tnode = DM.link(datasca.cl("Cost", "AN", "Stats",20,30));
        tnode.link(datasca.cl("<10", "DN", "",-20,10));
        tnode.link(datasca.cl("<1000", "DN", "",-10,10));
        tnode.link(datasca.cl("<10000", "DN", "",0,10));
        tnode.link(datasca.cl(">10000", "DN", "",10,10));

        tnode = datasca.find_node.named("Male");
        for (i=0; i<5; i++)
```

```
        create_DM_DN("Male", "Sports", "Summer", "<100");
        create_DM_DN("Male", "Sports", "Spring", "<1000");
        create_DM_DN("Male", "Appliances", "Fall", "<10000");
        create_DM_DN("Male", "Sports", "Summer", "<1000");

        for (i=0; i<5; i++)
            create_DM_DN("Female", "Clothes", "Fall", "<1000");
        create_DM_DN("Female", "Clothes", "Summer", "<100");
        create_DM_DN("Female", "Appliances", "Summer", "<1000");
        create_DM_DN("Female", "Appliances", "Winter", "<10000");
        *****/
```

```
    } // end popDM
```

```
    /**
     ** create_DM_DN
     **
     *****/
    public void create_DM_DN(String gen, String dept, String season, String cost) { // skip
        Node gender_node, dn;

        gender_node = datasca.find_node.named(gender);
        if (gender_node != null) {
            dn = gender_node.link(datasca.cl(""+global.counter++, "DN"));
            dn.link(dept); dn.link(season); dn.link(cost);
        }
    } // end create_DM_DN
    *****/
```

```
    /**
     ** con Populate Econ
     **
     */
    public void popEcon () { // skip
        Node Econ_node, tnode;

        GUI.P(0,"con", "Economist Run.");
        Econ_node = (DataSea.Root.link(datasca.cl("ECON", "AN", "Text",0,0)).link(datasca.cl("Economist This Week", "AN", "Text",0,10)));
        tnode = Econ_node.link(datasca.cl("ENTER NATO", "AN", "Text",0,20));
```

05/23/00  
00:16:59

## Confidential and Proprietary Property of Rocky Nevin

### Populate.java

10

```
(lnode.link(datasca.cl("NATO troops occupied KOSOVO after ")
+"Serb troops had agreed to withdraw", but found that a contingent of "
+"Russian soldiers had reached the airport first.", "Text", "Text", 0.300)).link(dat
asea.cl("NATO", "AN"));
(lnode=_lnode.link(datasca.cl("Despite some clashes between KLA fighters and Serbs,
)")
+"and between NATO soldiers and Serbs, NATO's takeover was largely peac
eful.", "Text", "Text", 1.20, 300)).link(datasca.cl("KLA", "AN"));
_lnode.link(datasca.find_node.named("NATO"));
*****
// end popEcon

/**
** popURLs
**
*/
public void popURLs() { // "show URLs"
Node URL0, URL1, URL2, URL3;

URL0 = create_node("URL:MSU-Bozeman Welcome", "URL");
URL0.Data[0] = "";
URL0.Data[1] = "Welcome to Montana State University at Bozeman!";
URL0.Data[2] = "";
URL0.Data[3] = "Below you will find topics of interest.";
URL0.Data[4] = "";
URL0.Data[5] = "For enrollment information for Spring 2001, wait here.";
URL0.Data[6] = "";
URL0.Data[7] = "- William Blake Archive ";
URL0.Data[8] = "- Contact Student Services";
URL0.Data[9] = "- Search Faculty/Staff Directory";

URL1 = create_node("URL:Netscape:Directories", "URL");
URL1.Data[0] = "The area code for all Montana locations is 406. ";
URL1.Data[1] = "The MSU-Bozeman campus operator can be reached at ";
URL1.Data[2] = "
(406)994-0211. ";
URL1.Data[3] = "";
URL1.Data[4] = "Montana State University ";
URL1.Data[5] = "";
URL1.Data[6] = "Montana State University-Bozeman ";
URL1.Data[7] = "- Faculty/Staff Directory ";
URL1.Data[8] = "- Student Directory ";
URL1.Data[9] = "- Department Mailing Addresses ";

URL2 = create_node("URL:MSU-Bozeman Faculty/Staff Directory ", "URL");
URL2.link(datasca.find_node.named("Dir"));
URL2.Data[0] = "";
URL2.Data[1] = "The faculty/staff directory is arranged alphabetically by last name.";
URL2.Data[2] = "";
URL2.Data[3] = "A B C D E F G H I J K L M N O P Q R S T U V W X Y Z ";
URL2.Data[4] = "Choose a letter and then utilize the search capability of your ";
URL2.Data[5] = "browser to locate specific individuals within the directory ";
URL2.Data[6] = "section. ";
URL2.Data[7] = "";
URL2.Data[8] = "Please send any directory additions or corrections to Barb Asper. ";
URL2.Data[9] = "This directory was last updated July 27, 1999. ";

URL3 = create_node("URL:MSU-Bozeman Faculty/Staff Directory (M)", "URL");
URL3.link(datasca.find_node.named("Dir"));
URL3.link(datasca.find_node.named("address"));
URL3.link(datasca.find_node.named("phone"));
URL3.link(datasca.find_node.named("email"));
URL3.Data[0] = "MSU-Bozeman Faculty/Staff Directory";
URL3.Data[1] = "MILLER, JOHN 278-7707";
URL3.Data[2] = "RES ASSOC, AES-WEST TRI AG RES";
URL3.Data[3] = "jpm@nervana.montana.edu";
URL3.Data[4] = "MILLER, JOHN 994-7332";
URL3.Data[5] = "DIRECTOR-PHD, COMPUTATIONAL BIOLOGY, AJ 29";
URL3.Data[6] = "jpm@nervana.montana.edu";
URL3.Data[7] = "MILLER, KATHERINE 994-5067";
URL3.Data[8] = "GRADUATE STUDENT, ENTOMOLOGY DEPARTMENT, LJ 415";
URL3.Data[9] = "kmliller@montana.edu";

URL0.link(datasca.find_node.named("Web"));
URL1.link(URL0);
URL2.link(URL1);
URL3.link(URL2);
URL3.link(triple("Miller", "(406)994-7332", "phone"));
triple("Miller", "jpm@nervana.montana.edu", "email");
triple("Miller", "123 Donnegal Dr, Bozeman, MT", "address");
(datasca.find_node.named("Miller")).link(datasca.find_node.named("name"));

URL2 = create_node("URL:The William Blake Archive", "URL");
URL2.Data[0] = "";
URL2.Data[1] = "Known Hazards and Most Favorable Conditions of the Archive ";
URL2.Data[2] = "";
URL2.Data[3] = "Graphical Help Screens (click icon for full-sized image)";
URL2.Data[4] = "";
URL2.Data[5] = "Help Table of Contents";
```

05/23/00  
00:16:59

## Confidential and Proprietary Property of Rocky Nevin

### Populate.java

11

```
URL2.Data[6] = "- The Archive and the Web. ";
URL2.Data[7] = "- Basic DynaWeb Navigation ";
URL2.Data[8] = "- Table of Contents";
URL2.Data[9] = "- Collection Index";

URL2.link(URL0);
(create_node("Blake", "DN"))link(datasca.cl("name", "AN"));
URL2.link(datasca.find_node.named("Blake"));

GUI.P(0,"popURLs", "show web");
return;

} // end popURLs

/**
 ** popl Populate time samples, 1,2,3 etc linked to dates, themselves linked to Timeline
 */
public void popl () {
    int x, y, i, size_X, size_Y;

    GUI.P(0,"popl", "time nodes Starting");

    popl0();

    datasca.gui.tl.create_TS(create_node("day1", "Event"), null, "Jan 1 1999");
    datasca.gui.tl.create_TS(create_node("day2", "Event"), null, "Feb 2 1999");
    datasca.gui.tl.create_TS(create_node("day3", "Event"), null, "Mar 3 1999");
    datasca.gui.tl.create_TS(create_node("day4", "Event"), null, "Apr 4 1999");
    datasca.gui.tl.create_TS(create_node("day5", "Event"), null, "May 5 1999");
    datasca.gui.tl.create_TS(create_node("day6", "Event"), null, "Jun 6 1999");
    datasca.gui.tl.create_TS(create_node("day7", "Event"), null, "Jul 7 2000");
    datasca.gui.tl.create_TS(create_node("day8", "Event"), null, "Aug 8 2000");
    datasca.gui.tl.create_TS(create_node("day9", "Event"), null, "Sep 9 2000");
    datasca.gui.tl.create_TS(create_node("day10", "Event"), null, "Oct 10 2000");
    datasca.gui.tl.create_TS(create_node("day11", "Event"), null, "Nov 11 2000");
    datasca.gui.tl.create_TS(create_node("day12", "Event"), null, "Dec 12 2000");

    GUI.P(0,"popl", "time nodes Done.");

    } // end popl

/**
 ** popTL Populate Timeline
 */
public void popTL () { // "show TL"
    int x, y, i, size_Y, size_X;
    int delta_x = 2; // offset from today_node, next week, etc. to TL

    GUI.P(0,"L", "Timeline Run.");
    // For demonstration purposes,
    // just code 'today', 'next-week', etc, as hard-coded dates

    //y=10;
    size_X=1;
    size_Y=(int)(Timeline.size_Y * 0.3);
    //x=(int)(Timeline.size_Y * 0.01);

    yesterday_node = new Node("Yesterday", "Event", "Yesterday");
    Node tn = datasca.gui.tl.create_TS(yesterday_node,null, "May 21 2000");
    yesterday_node.Y = tn.Y - 15;
    yesterday_node.X = Timeline.X - delta_x;
    yesterday_node.size_Y = 3;
    yesterday_node.size_X = 1;

    today_node = new Node("Today", "Event", "Today");
    tn = datasca.gui.tl.create_TS(today_node,null, "May 22 2000");
    today_node.Y = tn.Y;
    today_node.X = Timeline.X - delta_x;
    today_node.size_Y = 3;
    today_node.size_X = 1;

    tomorrow_node = new Node("Tomorrow", "Event", "Tomorrow");
    tn = datasca.gui.tl.create_TS(tomorrow_node,null, "May 23 2000");
    tomorrow_node.Y = tn.Y + 15;
    tomorrow_node.X = Timeline.X - delta_x;
    tomorrow_node.size_Y = 3;
    tomorrow_node.size_X = 1;

    last_week_node = new Node("LastWeek", "Event", "last_week");
    datasca.gui.tl.create_TS(last_week_node,null, "May 2 2000");
    datasca.gui.tl.create_TS(last_week_node,null, "May 3 2000");
    datasca.gui.tl.create_TS(last_week_node,null, "May 4 2000");
    datasca.gui.tl.create_TS(last_week_node,null, "May 5 2000");
    tn = datasca.gui.tl.create_TS(last_week_node,null, "May 6 2000");
    last_week_node.Y = tn.Y-60;
    last_week_node.X = Timeline.X - 2*delta_x;
    last_week_node.size_Y = 30;
    last_week_node.size_X = 1;
    datasca.gui.tl.create_TS(last_week_node,null, "May 7 2000");
```

05/23/00  
00:16:59

## Confidential and Proprietary Property of Rocky Nevin Populate.java

12

```
datasea.gui.tl.create_TS(last_week,node,null,"May 8 2000");
datasea.gui.tl.create_TS(last_week,node,null,"May 9 2000");

this_week.node = new Node("ThisWeek", "Event", "this_week");
```

```
datasea.gui.tl.create_TS(this_week,node,null,"May 22 2000");
datasea.gui.tl.create_TS(this_week,node,null,"May 23 2000");
datasea.gui.tl.create_TS(this_week,node,null,"May 24 2000");
datasea.gui.tl.create_TS(this_week,node,null,"May 25 2000");
tn = datasea.gui.tl.create_TS(this_week,node,null,"May 14 2000");
this_week.node.Y = tn.Y;
this_week.node.X = Timeline.X - 2*delta.x;
this_week.node.size.Y = 30;
this_week.node.size.X = 1;
datasea.gui.tl.create_TS(this_week,node,null,"May 26 2000");
datasea.gui.tl.create_TS(this_week,node,null,"May 27 2000");
datasea.gui.tl.create_TS(this_week,node,null,"May 28 2000");
```

```
next_week.node = new Node("NextWeek", "Event", "next_week");
```

```
datasea.gui.tl.create_TS(next_week,node,null,"May 29 2000");
datasea.gui.tl.create_TS(next_week,node,null,"May 30 2000");
datasea.gui.tl.create_TS(next_week,node,null,"May 31 2000");
datasea.gui.tl.create_TS(next_week,node,null,"Jun 1 2000");
tn = datasea.gui.tl.create_TS(next_week,node,null,"Mar 22 2000");
next_week.node.Y = tn.Y + 60;
next_week.node.X = Timeline.X - 2*delta.x;
next_week.node.size.Y = 30;
next_week.node.size.X = 1;
datasea.gui.tl.create_TS(next_week,node,null,"Jun 2 2000");
datasea.gui.tl.create_TS(next_week,node,null,"Jun 3 2000");
datasea.gui.tl.create_TS(next_week,node,null,"Jun 4 2000");
```

```
datasea.gui.tl.create_date_node("Feb 1 2000");
datasea.gui.tl.create_date_node("Apr 1 2000");
datasea.gui.tl.create_date_node("May 1 2000");
datasea.gui.tl.create_date_node("Jun 1 2000");
datasea.gui.tl.create_date_node("Jul 1 2000");
datasea.gui.tl.create_date_node("Aug 1 2000");
datasea.gui.tl.create_date_node("Sep 1 2000");
datasea.gui.tl.create_date_node("Oct 1 2000");
datasea.gui.tl.create_date_node("Nov 1 2000");
datasea.gui.tl.create_date_node("Dec 1 2000");
```

```
GUI.P0. "p00TL", "show TL";
```

```
return;
} // end TL
```

```
/**
** popcalendar
**
```

```
*/
```

```
public void popcalendar () {
```

```
int i, size;
```

```
Node child;
```

```
GUI.input.string_input("input Open House Sue&Fred Jan 1 2000");
```

```
GUI.input.string_input("input Bay Photo pickup Jan 3 2000");
```

```
GUI.input.string_input("input Bay Photo pickup Feb 3 2000");
```

```
GUI.input.string_input("input Custom Process Photo pickup Mar 30 2000");
```

```
GUI.input.string_input("input Bay Photo pickup Apr 23 2000");
```

```
GUI.input.string_input("input Darkroom 1 3-5pm Jan 15 2000");
```

```
GUI.input.string_input("input ASUC Photo exhibit reception 6-7pm Mar 24 2000");
```

```
GUI.input.string_input("input SOMArts exhibit reception 5-7pm Apr 6 2000");
```

```
GUI.input.string_input("input Tara & Gary BA 5:45pm Jan 7 2000");
```

```
GUI.input.string_input("input Lauren 2pm photo shoot Jan 20 2000");
```

```
GUI.input.string_input("input Lauren 1:30pm photo shoot Feb 8 2000");
```

```
GUI.input.string_input("input Dinner with Erin 7pm Mar 2 2000");
```

```
GUI.input.string_input("input Cal' Bach Society concert 9pm St Marks Mar 4 2000");
```

```
GUI.input.string_input("input Wat's master class SF Conservatory 11:30am Mar 6 2000");
```

```
GUI.input.string_input("input SFEMS Concert 8pm Mar 11 2000");
```

```
GUI.input.string_input("input SF FA Museum Floral exhibit Mar 14 2000");
```

```
GUI.input.string_input("input Sherman-Clay piano sale, UCB 1pm Mar 16 2000");
```

```
GUI.input.string_input("input Dinner with Erin 7pm Apr 9 2000");
```

```
} // end popcalendar
```

```
/**
```

```
** calendar
```

```
**
```

```
*/
```

```
public void calendar () {
```

```
int i,j,day=0;
```

```
Node cal,node, tnode;
```

```
cal.node = new Node("cal", "DN", "Calendar", 0, 0, 200, 150);
```

```
cal.node.set_mae(0.5);
```

05/23/00  
00:16:59

## Confidential and Proprietary Property of Rocky Nevin

### Populate.java

13

```
//cal_node.TinyScale = 0.1;
cal_node.link(Timeline);
datasea.set_child_position(Timeline, cal_node, 0, -300);

for (i=1; i<6; i++)
    for (j=1; j<8; j++) {
        if (++day > 31)
            break;
        tnode = new Node(""+day, "DN", "Day", 0, 0, 30, 20);
        tnode.link(cal_node);
        tnode.set_mag(0.9);
        datasea.set_child_position(cal_node, tnode, ((double)(j))/8.0, 0.1+((double)(5-i)
    ))/5.0);
    }

GUI.P(0, "popcalendar", "show cal");
} // end calendar

/*
**
*/
public void add_directory_entry (String input_string) {
    Node name=null, address=null, phone=null, other=null, email=null;
    Node Directory, name_an_node, address_an_node, phone_an_node, email_an_node;
    int n in_words, it;
    String words[];

    Node DirCNode = create_node("DirCN", "CN");
    Node MailDirCNode = create_node("MailDirCN", "CN");

    StringTokenizer t = new StringTokenizer(input_string, ".,<>\\\"'\\n");
    num_words = t.countTokens();
    words = new String[num_words];
    for (i = 0; i < num_words; i++)
        words[i] = t.nextToken();

    Directory = datasea.find_node_named("Dir");
    if (Directory==null) {
        Directory = new Node("Dir", "AN", "");
        DataSea.Root.link(Directory, "polarized");
        GUI.P(1, "add_directory_entry", "Creating Directory, linking it to DataSea.Root");
    }
    name_an_node = datasea.find_node_named("name");
    if (name_an_node==null) {
        name_an_node = new Node("name", "AN", "");
        Directory.link(name_an_node, DirCNode, "polarized");
        address_an_node = datasea.find_node_named("address");
        if (address_an_node==null) {
            address_an_node = new Node("address", "AN", "");
            Directory.link(address_an_node, DirCNode, "polarized");
            phone_an_node = datasea.find_node_named("phone");
            if (phone_an_node==null) {
                phone_an_node = new Node("phone", "AN", "");
                //Node telephone_an_node = new Node("telephone", "AN", "");
                //telephone_an_node.link(phone_an_node);
                Directory.link(phone_an_node, DirCNode, "polarized");
                email_an_node = datasea.find_node_named("email");
                if (email_an_node==null) {
                    email_an_node = new Node("email", "AN", "");
                    Directory.link(email_an_node, MailDirCNode, "polarized");
                }
            }
        }
    }
    if (num_words >= 1) {
        name = datasea.find_node_named(words[0]);
        if (name==null)
            name=new Node(words[0], "DN", "name");
        name_an_node.link(name, "polarized");
        //name_an_node.link(name, DirCNode, "polarized");
    }
    if (num_words >= 2) {
        phone = datasea.find_node_named(words[1]);
        if (phone==null)
            phone=new Node(words[1], "DN", "phone");
        phone_an_node.link(phone, "polarized");
        //phone_an_node.link(phone, DirCNode, "polarized");
    }
    if (num_words >= 3) {
        address = datasea.find_node_named(words[2]);
        if (address==null)
            address=new Node(words[2], "DN", "address");
        address_an_node.link(address, "polarized");
        //address_an_node.link(address, DirCNode, "polarized");
    }
}
```

05/23/00  
00:16:59

Confidential and Proprietary Property of Rocky Nevin  
Populate.java

14

```
if (num_words >= 4) {
    email = datasea.find_node.named(words[3]);
    if (email==null)
        email=new Node(words[3], "DN", "email");
    email_an.node.link(email, MailDirCNode, "polarized");
}

name.link(phone, DirCNode, "unpolarized");
name.link(address, DirCNode, "unpolarized");
name.link(email, DirCNode, "unpolarized");

datasea.gui.show_node.once(name);
datasea.gui.show_node.once(phone);
datasea.gui.show_node.once(address);
datasea.gui.show_node.once(email);

//datasea.set_CSs_of_DirCNode(DirCNode, 10);
//datasea.set_CSs_of_DirCNode(MailDirCNode, 30);

} // end add_directory_entry

/**
 ** DIRECTORY
 **
 */
public void popd () { // "show Dir"
    Node tn;

    add_directory_entry("Abe 111-1111 1111, Abner abe@aol.com");
    add_directory_entry("Bob 222-2222 2222, Broadway bob@aol.com");
    add_directory_entry("Carl 333-3333 1234, Claremont carl@aol.com");
    add_directory_entry("Dave 444-4444 1234, Denison dave@aol.com");
    add_directory_entry("Evan 555-5555 1234, Edgar evan@aol.com");
    add_directory_entry("Frank 666-6666 1234, Fairway frank@aol.com");
    add_directory_entry("Rocky 888-8888 1234, Campus rocky@talis.com");
}
```

```
/**
 ** popportal
 **
 *****
 public void popportal () { // "show Yahoo"
     Node n1, n2, n3, n4, n5, n6, n7, n8;
     System.err.println("Populate.portal() begun");

     Node Yahoo = create_node("Yahoo", "AN");
     ANpair("Yahoo", "Animal");
     ANpair("Animal", "FoodAnimals");
     ANpair("FoodAnimals", "Poultry");
     ANpair("FoodAnimals", "Cattle");
     ANpair("Animal", "Reproduction");
     ANpair("Reproduction", "Egg");
     ANpair("Egg", "EggTempera");
     ANpair("Egg", "Reproduction");
     ANpair("Egg", "Poultry");
     ANpair("Animal", "Reproduction");
     ANpair("EggTempera", "ETRecipes");
     ANpair("EggTempera", "Longevity");
     ANpair("EggTempera", "Longevity");
     ANpair("EggTempera", "OldMaster_Techniques");
     ANpair("EggTempera", "Techniques");
     ANpair("EggTempera", "ArtMedia");
     ANpair("Animal", "Morphology");
     ANpair("Animal", "Morphology");
     ANpair("Morphology", "Skeletal");
     ANpair("Morphology", "Circulation");
     ANpair("Morphology", "Skin");
     ANpair("Morphology", "Morphology");
     ANpair("Yahoo", "Art");
     ANpair("Art", "ArtMaterials");
     ANpair("Art", "ArtHistory");
     ANpair("Art", "Artists");
     ANpair("Art", "Painting");
     ANpair("Animal", "Environment");
     ANpair("ArtMaterials", "ArtMedia");
     ANpair("Painting", "Fresco");
     ANpair("Painting", "OilPaintings");
     ANpair("Painting", "WaterColors");
     ANpair("Fresco", "EggTempera");
     ANDNpair("Poultry", "EggProduction");
 }
```

```
ANDNpair("Egg", "EggProduction");

ANDNpair("ArtMedia", "Artmedia&Fresco");
ANDNpair("Fresco", "Artmedia&Fresco");

ANDNpair("EggTempera", "ET&Fresco");
ANDNpair("Fresco", "ET&Fresco");

ANDNpair("OilPaintings", "OldMasterPaintings");
ANDNpair("Fresco", "OldMasterPaintings");

ANDNpair("Egg", "Egg&Skin");
ANDNpair("Skin", "Egg&Skin");

ANDNpair("Morphology", "Skin&Morph");
ANDNpair("Skin", "Skin&Morph");

ANDNpair("Reproduction", "EggIncubation");
ANDNpair("Egg", "EggIncubation");
ANDNpair("Egg", "EggNutrition");
ANDNpair("FoodAnimals", "EggNutrition");
ANDNpair("Poultry", "EggNutrition");
ANDNpair("Reproduction", "LiveBirth");

ANDNpair("Animal", "Animal&Morph");
ANDNpair("Morphology", "Animal&Morph");

////////// search result node linked to 100 URLs
////////// group of 100 URLs
////////// ANs distal
////////// user selects ANs, samples URLs, adds more search terms +/-

GUI.P(0, "popportal", "show Yahoo");

popmany();

System.err.println("Populate portal() done.");
System.err.println("Populate portal() Yahoo =" + Yahoo);
} // end popportal
```

```
/**
** doomnav like doportal. but with more links to each DN
```

```

**
public void popmany () { // "show Yahoo"
    Node n1, n2, n3, n4, n5, n6, n7, n8;
    // Node Art, ArtMaterials, Frescoe, EggTempera OldMaster_Techniques;
    // Node Egg, Animal, Reproduction, Morphology, Circulation, Skeletal, Skin;

    Node Yahoo = create_node("xYahoo", "AN");
    Node Egg = create_node("xEgg", "AN");
    Node Animal = create_node("xAnimal", "AN");
    Node Reproduction = create_node("xReproduction", "AN");
    Node Morphology = create_node("xMorphology", "AN");
    Node Circulation = create_node("xCirculation", "AN");
    Node Skeletal = create_node("xSkeletal", "AN");
    Node Skin = create_node("xSkin", "AN");
    Node Art = create_node("xArt", "AN");
    Node ArtMaterials = create_node("xArtMaterials", "AN");
    Node Frescoe = create_node("xFrescoe", "AN");
    Node EggTempera = create_node("xEggTempera", "AN");
    Node OldMaster_Techniques = create_node("xOldMaster_Techniques", "AN");
    Node ArtMedia = create_node("xArtMedia", "AN");
    Node Recipes = create_node("xRecipes", "AN");
    Node Longevity = create_node("xLongevity", "AN");
    Node Techniques = create_node("xTechniques", "AN");
    Node Environment = create_node("xEnvironment", "AN");
    Node LiveBirth = create_node("xLiveBirth", "AN");
    Node Poultry = create_node("xPoultry", "AN");
    Node ArtHistory = create_node("xArtHistory", "AN");
    Node Artists = create_node("xArtists", "AN");

    Yahoo.link(Animal, "polarized");
    Yahoo.link(Art, "polarized");
    Egg.link(EggTempera, "polarized");
    Egg.link(Reproduction, "polarized");
    Egg.link(Poultry, "polarized");
    Animal.link(Reproduction, "polarized");
    Animal.link(Environment, "polarized");
    Animal.link(Morphology, "polarized");
    EggTempera.link(Recipes, "polarized");
    EggTempera.link(Longevity, "polarized");
    EggTempera.link(OldMaster_Techniques, "polarized");
    EggTempera.link(Techniques, "polarized");
    EggTempera.link(ArtMedia, "polarized");
    Morphology.link(Skeletal, "polarized");
    Morphology.link(Circulation, "polarized");
    Morphology.link(Skin, "polarized");
```

05/23/00  
00:16:59

## Confidential and Proprietary Property of Rocky Nevin

Populate.java

16

```
Reproduction.link(LiveBirth, "polarized");
Art.link(ArtMaterials, "polarized");
Art.link(ArtHistory, "polarized");
Art.link(Artists, "polarized");
ArtMaterials.link(ArtMedia, "polarized");
Fresco.link(EggTempera, "polarized");
```

```
////////// search result node linked to 100 URLs
////////// group of 100 URLs
////////// AN's distal
//////////
////////// user selects AN's, samples URLs, adds more search terms +/-
//////////
```

```
int counter=0;
Node dn = create_node("xArtmedia&Fresco", "DN");
ArtMedia.link(dn);
Fresco.link(dn);
Art.link(dn);
EggTempera.link(dn);
```

```
dn = create_node("xET&Fresco", "DN");
ArtMedia.link(dn, "polarized");
Fresco.link(dn, "polarized");
Art.link(dn, "polarized");
EggTempera.link(dn, "polarized");
```

```
dn = create_node("xEgg&Skin", "DN");
Egg.link(dn, "polarized");
Skin.link(dn, "polarized");
LiveBirth.link(dn, "polarized");
Reproduction.link(dn, "polarized");
```

```
dn = create_node("xSkin&Morph", "DN");
Morphology.link(dn, "polarized");
Skin.link(dn, "polarized");
```

```
dn = create_node("xAnimal&Morph", "DN");
Animal.link(dn, "polarized");
Morphology.link(dn, "polarized");
Environment.link(dn, "polarized");
Reproduction.link(dn, "polarized");
```

```
GUI.P("popmany", "show xYahoo");
```

```
} // end popmany
*****
```

```
/**
** popframe Christmas-tree frame for other data to attach to
**
*/
```

```
public void popframe () { // "show n1"
Node n1, n2, n3, n4, n5, n6, n7, n8;
//String type = "BB";
String type = "DN";
```

```
n1 = create_node("n1", type);
n1.X=40;
n1.Y=50;
dataseta.Root.link(n1);
```

```
////////// VERTICAL PIECES
```

```
n2 = create_node("n2", type);
n2.link(n1);
```

```
n2.X=0;
n2.Y = 75;
```

```
n3 = create_node("n3", type);
n3.link(n2);
```

```
n3.X=0;
n3.Y = 75;
```

```
//////////
```

```
////////// HORIZONTAL PIECES
```

```
n4 = create_node("n4", type);
```

```
n4.link(n3);
```

```
n4.X=50;
```

```
n4.Y = -25;
```

```
n5 = create_node("n5", type);
```

```
n5.link(n3);
```

```
n5.X=-50;
```

```
n5.Y = -25;
```

```
n6 = create_node("n6", type);
```

```
n6.link(n2);
```

```
n6.X=50;
```

```
n6.Y = -25;
```

```
n7 = create_node("n7", type);
```

05/23/00  
00:16:59

# Confidential and Proprietary Property of Rocky Nevin Populate.java

17

```
n7.link(n2);
n7.X=-50;
n7.Y=-25;

GUI.P(0, "popframe", "show n1");
return;
} // end popframe

/**
** popp
**
** creates mail,url and note, links them to lastweek,yesterday and today
*/
public void popp () {
    int i, size;
    Node child;

    // Mary's EMAIL, about printer
    Node mail = email("Mary EMAIL, I really like HP Printer SMP");
    dataasea.gui.il.create_TS(mail,null,"Jul 3 2000");

    // Bob's NOTE, about printer 'A' in Joe Baker's Office
    Node n = GUI.input.string_input("input Bob said it would be $300 to fix Joe Baker Printer named
    'A'");

    // URL advertisement for HP printer
    Node url_node = create_node("Ad for HP Printer SMP", "URL");
    url_node.Desc="HP Advertisement";
    url_node.link(dataasea.find_node.named("Printer"));
    url_node.link(dataasea.find_node.named("SMP"));
    url_node.link(dataasea.find_node.named("HP"));
    // url_node.link(dataasea.find_node.named("Yesterday"));
    dataasea.gui.il.create_TS(url_node,null,"Jul 9 2000");

    //popp();

    return;
} // end popp

/**
** poppp
**
** creates mail,url and note, links them to lastweek,yesterday and today
*/
public void poppp () {
    int i, size;
    Node child;

    // Blah EMAIL
    email("Bob EMAIL, This is an interesting day, my cat is happy");
    email("Bob EMAIL, Yesterday is an interesting day, my dog is happy");
    email("Bob EMAIL, Tomorrow is an interesting day, my parrot is happy");
    email("Bob EMAIL, LastWeek was an interesting week, my cougar is happy");

    // Blah URL
    Node url_node = create_node("kjkajdf kjasdFkjaskdf", "URL");
    url_node.Desc="URL, Blah blah";
    url_node.link(dataasea.find_node.named("blah"));

    // Blah URL
    url_node = create_node("kjksa asdkjaskf kasf", "URL");
    url_node.Desc="URL, akjaskdf kasdf lasdf something";
    url_node.link(dataasea.find_node.named("LastWeek"));

    // Blah URL
    url_node = create_node("ak jkska l aslas dfs", "URL");
    url_node.Desc="URL, xxxk asdkf asdfas";
    url_node.link(dataasea.find_node.named("xxxk"));
    url_node.link(dataasea.find_node.named("Tomorrow"));

    // Blah Notes ...
    GUI.input.string_input("input Abe believes in flying saucers, Jan 5 1999");
    GUI.input.string_input("input Abe thinks of nothing, Feb 10 1999");
    GUI.input.string_input("input Bob says everything, Mar 20 1999");
    GUI.input.string_input("input Bob drives a car, Jan 25 2000");
    GUI.input.string_input("input Carl does nothing, Mar 30 2000");

    return;
} // end poppp

/**
** email
**
** public Node email (String input_string) {
    int i, size;
    Node child;
    String individual word;
```

05/23/00  
00:16:59

## Confidential and Proprietary Property of Rocky Nevlin

### Populate.java

18

```
Node email_node = create_node("email");
Node mail = create_node(input_string, "DN");

String useful_string = dataseta.gui.input_discard_words(input_string);
StringTokenizer t = new StringTokenizer(useful_string, " ");
int num_words = t.countTokens();

for(i = 0; i < num_words; i++) {
    individual_word = t.nextToken();
    mail.link(create_node(individual_word, "DN"));
}

return(mail);
} // end email

/**
 ** popnet
 **
 */
public void popnet () {
    int i, size;
    Node child;

    pull_in_URLs("http://www.paintedeggs.com/", "eggs2.html", 3, (Node)null);
    pull_in_URLs("http://www.alvr.com/", "eggs2mainpage.html", 3, (Node)null);
    /*****
    pull_in_URLs("http://www.eggtempera.com/", "faq.html", 1, (Node)null);
    pull_in_URLs("http://netvet.wustl.edu", "birds.htm", 1, (Node)null);
    pull_in_URLs("http://www.berkeley.edu", "index.html", 1, (Node)null);
    *****/
    /*****
    pull_in_URLs("http://www.eggtempera.com/", "welcome.html", 1, (Node)null);
    pull_in_URLs("http://www.eggtempera.com/", "stipotions.html", 1, (Node)null);
    pull_in_URLs("http://www.eggtempera.com/", "aboutsociety.html", 1, (Node)null);
    pull_in_URLs("http://www.eggtempera.com/", "aboutet.html", 1, (Node)null);
    pull_in_URLs("http://www.eggtempera.com/", "history.html", 1, (Node)null);
    pull_in_URLs("http://www.eggtempera.com/", "publications.html", 1, (Node)null);
    pull_in_URLs("http://www.eggtempera.com/", "instructors.html", 1, (Node)null);
    pull_in_URLs("http://www.eggtempera.com/", "materials.html", 1, (Node)null);
    pull_in_URLs("http://www.eggtempera.com/", "guestbook.html", 1, (Node)null);
    pull_in_URLs("http://www.eggtempera.com/", "links.html", 1, (Node)null);
    pull_in_URLs("http://www.eggtempera.com/", "intro.html", 1, (Node)null);
    *****/

    } // end popnet

    /**
    ** popweb sample of web history
    **
    */
    public void popweb () { // "show WebHistory"
        Node l, m;
        int counter = 0;
        int counter_delta=2;

        l = create_node("WebHistory", "DN");
        l.X=20;
        l.Y=counter-counter_delta;
        dataseta.Root.link(l);
        l.link(dataseta.find_node_named("Web"));

        m = create_node("Search:EggTempera", "DN");
        m.X=20;
        m.Y=counter-counter_delta;
        l.link(m);

        Node n1 = create_node("http://www.eggtempera.com/welcome.html", "URL");
        Node n2 = create_node("http://www.eggtempera.com/stipotions.html", "URL");
        Node n3 = create_node("http://www.eggtempera.com/aboutsociety.html", "URL");
        Node n4 = create_node("http://www.eggtempera.com/aboutet.html", "URL");
        Node n5 = create_node("http://www.eggtempera.com/history.html", "URL");
        Node n6 = create_node("http://www.eggtempera.com/publications.html", "URL");
        Node n7 = create_node("http://www.eggtempera.com/instructors.html", "URL");
        Node n8 = create_node("http://www.eggtempera.com/faq.html", "URL");
        Node n9 = create_node("http://www.eggtempera.com/materials.html", "URL");
        Node n10 = create_node("http://www.eggtempera.com/guestbook.html", "URL");
        Node n11 = create_node("http://www.eggtempera.com/links.html", "URL");
        Node n12 = create_node("http://www.eggtempera.com/intro.html", "URL");

        m.link(n1);
        n1.link(n2);
        n2.link(n3);
        n3.link(n4);
        n4.link(n5);
        n5.link(n6);
        n6.link(n7);
        n7.link(n8);
        n8.link(n9);
        n9.link(n10);
        n10.link(n11);
    }
```

```
n1.l.link(n12);
```

```
n5.l.link(datasca.find_node_named("x5"));
n7.l.link(datasca.find_node_named("x7"));
```

```
n1.X=0; n1.Y=counter==counter_delta;
n2.X=0; n2.Y=counter==counter_delta;
n3.X=0; n3.Y=counter==counter_delta;
n4.X=0; n4.Y=counter==counter_delta;
n5.X=0; n5.Y=counter==counter_delta;
n6.X=0; n6.Y=counter==counter_delta;
n7.X=0; n7.Y=counter==counter_delta;
n8.X=0; n8.Y=counter==counter_delta;
n9.X=0; n9.Y=counter==counter_delta;
n10.X=0; n10.Y=counter==counter_delta;
n11.X=0; n11.Y=counter==counter_delta;
n12.X=0; n12.Y=counter==counter_delta;
```

```
Node cn = create_node("WebSession.1", "CN");
```

```
cn.l.link(n1);
cn.l.link(n2);
cn.l.link(n3);
cn.l.link(n4);
cn.l.link(n5);
cn.l.link(n6);
cn.l.link(n7);
cn.l.link(n8);
cn.l.link(n9);
cn.l.link(n10);
cn.l.link(n11);
cn.l.link(n12);
//cn.l.link(datasca.find_node_named("today"));
```

```
GUI.P(0,"popweb", "show WebHistory");
```

```
return;
```

```
} // end popweb
```

```
/**
** poprev
**
*/
public void poprev () {
int i, size;
```

```
Node child;
```

```
Node ruth = create_node("Ruth");
Node A1 = create_node("A1");
Node A2 = create_node("A2");
Node A3 = create_node("A3");
Node A4 = create_node("A4");
Node target = create_node("target");
Node B1 = create_node("B1");
```

```
ruth.l.link(A1);
A1.l.link(A2);
A2.l.link(A3);
A3.l.link(A4);
A4.l.link(target);
ruth.l.link(B1);
B1.l.link(A4);
```

```
// ruth - A1 - A2 - A3 - A4 - target
// \---B1-----/
// set_Tdist_start(ruth, target) should order them thus:
//
// 1 - 2 - 3 - 4 - 5 - 6
// \ 2 -----/
```

```
} // end poprev
```

```
/**
```

```
** popegg
```

```
**
```

```
*/
public void popegg () {
```

```
int i, size;
```

```
Node child;
```

```
pull_in_URLs("http://www.eggtempera.com/", "welcome.html", 1, (Node)null);
pull_in_URLs("http://www.eggtempera.com/", "stipotions.html", 1, (Node)null);
pull_in_URLs("http://www.eggtempera.com/", "aboutsociety.html", 1, (Node)null);
pull_in_URLs("http://www.eggtempera.com/", "aboutel.html", 1, (Node)null);
pull_in_URLs("http://www.eggtempera.com/", "history.html", 1, (Node)null);
pull_in_URLs("http://www.eggtempera.com/", "publications.html", 1, (Node)null);
pull_in_URLs("http://www.eggtempera.com/", "instructors.html", 1, (Node)null);
pull_in_URLs("http://www.eggtempera.com/", "faq.html", 1, (Node)null);
pull_in_URLs("http://www.eggtempera.com/", "materials.html", 1, (Node)null);
pull_in_URLs("http://www.eggtempera.com/", "questbook.html", 1, (Node)null);
```

```

pull_in_URLs("http://www.eggtempera.com/", "links.html", 1, (Node)null);
pull_in_URLs("http://www.eggtempera.com/", "intro.html", 1, (Node)null);
} // end popegg

/**
 ** popBB Backbone network
 **
 */
public void popBB () { // "show Lessons"
    Node l, m, prior, x, y, z, book_node;
    int counter = -30;
    int counter_delta = 3;

    Node CN1 = new Node("BackBoneCN", "CN");

    l = create_node("Lessons", "DN");
    l.Y = counter += 10;
    l.X = 20;
    datasea.Root.link(l);

    l.link(make_BB("start", "Lesson1"));
    prior = lastBBnode;
    lastBBnode.Y = counter += counter_delta;
    lastBBnode.X = 20;
    make_BB("add", "Chapter_1");
    prior = lastBBnode;
    CN1.link(prior);
    lastBBnode.Y = counter += counter_delta;
    lastBBnode.X = 20;
    make_BB("add", "Chapter_2");
    prior.link(book_node = create_node("Some Book1", "DN"));
    CN1.link(prior);
    x = create_node("x1", "DN");
    y = create_node("y1", "DN");
    z = create_node("z1", "DN");
    book_node.link(x);
    x.link(y);
    y.link(z);
    prior = lastBBnode;
    lastBBnode.Y = counter += counter_delta;
    lastBBnode.X = 20;

    make_BB("add", "Chapter_3");
    prior.link(book_node = create_node("Some Book2", "DN"));
    CN1.link(prior);
    x = create_node("x2", "DN");
    y = create_node("y", "DN");
    z = create_node("z2", "DN");
    book_node.link(x);
    x.link(y);
    y.link(z);
    prior = lastBBnode;
    lastBBnode.Y = counter += counter_delta;
    lastBBnode.X = 20;
    make_BB("add", "Chapter_4");
    prior.link(book_node = create_node("Some Book3", "DN"));
    CN1.link(prior);
    x = create_node("x3", "DN");
    y = create_node("y3", "DN");
    z = create_node("z3", "DN");
    book_node.link(x);
    x.link(y);
    y.link(z);
    prior = lastBBnode;
    lastBBnode.Y = counter += counter_delta;
    lastBBnode.X = 20;
    make_BB("add", "Chapter_5");
    prior.link(book_node = create_node("Some Book4", "DN"));
    CN1.link(prior);
    x = create_node("x4", "DN");
    y = create_node("y4", "DN");
    z = create_node("z4", "DN");
    book_node.link(x);
    x.link(y);
    y.link(z);
    prior = lastBBnode;
    lastBBnode.Y = counter += counter_delta;
    lastBBnode.X = 20;
    make_BB("add", "Chapter_6");
    prior.link(book_node = create_node("Some Book5", "DN"));
    CN1.link(prior);
    x = create_node("x5", "DN");
    y = create_node("y5", "DN");
    z = create_node("z5", "DN");
    book_node.link(x);
    x.link(y);
    y.link(z);
    prior = lastBBnode;

```

```
lastBNode.Y = counter+=counter_delta;
lastBNode.X=20;
make_BB("add", "Chapter.7");
prior.link(book_node = create_node("Some Book6", "DN"));
CN1.link(prior);
x=create_node("x6", "DN");
y=create_node("y6", "DN");
z=create_node("z6", "DN");
book_node.link(x);
x.link(y);
y.link(z);
prior = lastBNode;
lastBNode.Y = counter+=counter_delta;
lastBNode.X=20;
make_BB("add", "Chapter.8");
prior.link(book_node = create_node("Some Book7", "DN"));
CN1.link(prior);
x=create_node("x7", "DN");
y=create_node("y7", "DN");
z=create_node("z7", "DN");
book_node.link(x);
x.link(y);
y.link(z);
prior = lastBNode;
lastBNode.Y = counter+=counter_delta;
lastBNode.X=20;
make_BB("add", "Chapter.9");
prior.link(book_node = create_node("Some Book8", "DN"));
CN1.link(prior);
x=create_node("x8", "DN");
y=create_node("y8", "DN");
z=create_node("z8", "DN");
book_node.link(x);
x.link(y);
y.link(z);
prior = lastBNode;
lastBNode.link(book_node = create_node("Some Book9", "DN"));
lastBNode.Y = counter+=counter_delta;
lastBNode.X=20;

Node CN2 = new Node("BackBoneCN", "CN");
link(make_BB("start", "Lesson2"));
prior = lastBNode;
lastBNode.Y = counter+=counter_delta;
lastBNode.X=20;
make_BB("add", "xChapter.1");

prior = lastBNode;
lastBNode.Y = counter+=counter_delta;
lastBNode.X=20;
make_BB("add", "xChapter.2");
prior.link(book_node = create_node_forced("Another Book1", "DN"));
CN2.link(prior);
x=create_node_forced("xx1", "DN");
y=create_node_forced("yy1", "DN");
z=create_node_forced("zz1", "DN");
book_node.link(x);
x.link(y);
y.link(z);
prior = lastBNode;
lastBNode.Y = counter+=counter_delta;
lastBNode.X=20;
make_BB("add", "xChapter.3");
prior.link(book_node = create_node_forced("Another Book2", "DN"));
CN2.link(prior);
x=create_node_forced("xx2", "DN");
y=create_node_forced("yy", "DN");
z=create_node_forced("zz2", "DN");
book_node.link(x);
x.link(y);
y.link(z);
prior = lastBNode;
lastBNode.Y = counter+=counter_delta;
lastBNode.X=20;
make_BB("add", "xChapter.4");
prior.link(book_node = create_node_forced("Another Book3", "DN"));
CN2.link(prior);
x=create_node_forced("xx3", "DN");
y=create_node_forced("yy3", "DN");
z=create_node_forced("zz3", "DN");
book_node.link(x);
x.link(y);
y.link(z);
prior = lastBNode;
lastBNode.Y = counter+=counter_delta;
lastBNode.X=20;
make_BB("add", "xChapter.5");
prior.link(book_node = create_node_forced("Another Book4", "DN"));
CN2.link(prior);
x=create_node_forced("xx4", "DN");
y=create_node_forced("yy4", "DN");
z=create_node_forced("zz4", "DN");
book_node.link(x);
```

```

z=create_node_forced("zz8", "DN");
book_node.link(x);
x.link(y);
y.link(z);
lastBNode.link(book_node = create_node_forced("Another Book9", "DN"));
lastBNode.Y = counter+=counter_delta;
lastBNode.X=20;

GUI.P(0, "popBB", "show Lessons");
} // end popBB

/**
** pops SIMILAR NODES
**
*/

public void pops () { // "show egg-tempera"
Node node.sim;
Node et, gt, si, f, food, acrylics, oils;
Node AP, PT, E, RP, RI;

GUI.P(0, "", "late similarities, e.g. Egg-Tempera");
//
node.sim=create_node("sim", "AN", "");

et=create_node("Egg-Tempera", "DN", "");
gt=create_node("Glazing Techniques", "DN", "");
si=create_node("Secular Images", "DN", "");
f=create_node("Frescoes", "DN", "");
food=create_node("food", "DN", "");
acrylics=create_node("acrylics", "DN", "");
oils=create_node("oils", "DN", "");

AP=create_node("Art Practice", "AN", "");
PT=create_node("Painting Techniques", "AN", "");
E=create_node("Eggs", "AN", "");
RP=create_node("Renaissance-Paintings", "AN", "");
RI=create_node("Religious Images", "AN", "");

AP.link(et);
PT.link(et);
E.link(et);
RP.link(et);
RI.link(et);

AP.link(gt);
PT.link(et);

```

```
AP.link(0);
PT.link(0);
AP.link(acrylics);
PT.link(acrylics);
AP.link(oils);
PT.link(oils);
E.link(food);
RP.link(si);
RP.link(0);
RI.link(0);
```

```
datasea.needsToUpdate = true;
GUI.P(0, "pops", "show egg-tempera");
return;
} // end (SIMILAR NODES)
```

```
/**
 ** popm MAIL
 **
 */
public void popm () { // "show email"
int i;
Node email, read, unread;
```

```
DataSea.currentCNode = create_node("MailCN", "CN"); // Turn on auto-CNode links
email("Hi Abe, this is email written about dogs.");
email("Dear Mary, How's the project going?");
email("Bob: please turn in your budget for next year.");
email("Carl: don't forget to close the door.");
```

```
DataSea.currentCNode = null; // Turn off auto-CNode links
datasea.needsToUpdate = true;
GUI.P(0, "popm", "show email");
return;
} // end (pop-email)
```

```
/**
 ** get_node_from_file_suffix
 **
 */
public Node get_node_from_file_suffix (String file_name) {
```

```
int index;
Node ret_file=null;
```

```
/******
```

```
Node Programming = create_node("Programming", "AN");
```

```
Programming.link(JAVA);
```

```
Programming.link(CLASS);
```

```
Node Text = create_node("Text", "AN");
```

```
Text.link(DOC);
```

```
Text.link(TXT);
```

```
Node Images = create_node("Images", "AN");
```

```
Images.link(JPG);
```

```
Images.link(PS);
```

```
*****
```

```
if (0 <= (index = file_name.toLowerCase().indexOf(".java"))) {
ret_file = create_node("JAVA", "CN");
}
```

```
else
if (0 <= (index = file_name.toLowerCase().indexOf(".class"))) {
ret_file = create_node("CLASS", "CN");
}
```

```
else
if (0 <= (index = file_name.toLowerCase().indexOf(".jpg"))) {
ret_file = create_node("JPG", "CN");
}
```

```
else
if (0 <= (index = file_name.toLowerCase().indexOf(".ps"))) {
ret_file = create_node("PS", "CN");
}
```

```
else
if (0 <= (index = file_name.toLowerCase().indexOf(".doc"))) {
ret_file = create_node("DOC", "CN");
}
```

```
else
if (0 <= (index = file_name.toLowerCase().indexOf(".txt"))) {
ret_file = create_node("TXT", "CN");
}
```

```
else
if (0 <= (index = file_name.toLowerCase().indexOf(".html"))) {
ret_file = create_node("HTML", "CN");
}
```

```
else
if (0 <= (index = file_name.toLowerCase().indexOf(".gif"))) {
```

```

        rel_file = create_node("GIF", "CN");
    }
}

else
if (0 <= (index = file_name.toLowerCase().indexOf("c"))) {
    rel_file = create_node("C", "CN");
}
return(rel_file);
} // end get_node_from_file_suffix

/**
 ** popf FILES
 **
 */
public void popf () { // "show Files or Files"
    Node file_node=null;

    create_file_tree((File)null, 0);

    file_node = datasea.find_node_named("Files");
    datasea.clean(file_node);

    GUI.P(0, "popf", "show Files or Files");
    return;
} // end popf

/**
 ** create_file_tree
 **
 */
public Node create_file_tree (File this_file, int current_depth) {
    int i, max_length, index=0;
    String list[], tName, path;
    File tFile;
    // user_dir_file=null; //user_dir_file is used temporarily only
    char separator;
    long mod, this_file.mod;
    Node this_file_node, tFile_node, FilesNode=null;
    boolean starting = false;

    if (current_depth > MAX_DIRECTORY_DEPTH) // limit the depth of recursion
        return((Node)null);

    FilesNode = create_node("Files", "AN");

    if (this_file == null) { // if starting, link to DataSea.Root
        starting = true;
        GUI.P(0, "create_file_tree", "starting on user:home <"+GUI.GlobalUserHomeDir+">");
        this_file = new File(GUI.GlobalUserHomeDir); // Start from UserDir,
    }
    try {
        path = this_file.getCanonicalPath();
        GUI.P(0, "create_file_tree", "getCanonicalPath()="+path);
    }
    catch (IOException e) {
        GUI.P(0, "create_file_tree", "getCanonicalPath(): "+e);
    }
}

this_file_node = create_node.forced(this_file.getName(), "DN", ""); // force creation
this_file_node.isFile = true;

this_file.mod = this_file.lastModified();
separator = this_file.separatorChar;
path = this_file.getPath();

/*****
GUI.P(0, "create_file_tree", "path="+path);
GUI.P(0, "create_file_tree", "pathSeparator="+this_file.pathSeparator);
GUI.P(0, "create_file_tree", "separator="+this_file.separator);
GUI.P(0, "create_file_tree", "Name="+this_file.getName());
GUI.P(0, "create_file_tree", "Parent="+this_file.getParent());
GUI.P(0, "create_file_tree", "isDirectory="+this_file.isDirectory());
GUI.P(0, "create_file_tree", "=====");
GUI.P(0, "create_file_tree", "=====");
*****/

if (this_file.isDirectory()) {
    this_file_node.isDirectory = true;
    GUI.P(0, "create_file_tree", "this File <"+this_file.getName()+"> is a directory.");
    list = this_file.list();
    if (list == null) {
        GUI.WARNING(0, "create_file_tree", "list is null, returning.");
        return((Node)null);
    }
    max_length = (list.length > MAX_FILES_PER_DIRECTORY ? (MAX_FILES_PER_DIRECTORY) : (list.length));
    for (i=0; i< max_length; i++) {

```

05/23/00  
00:16:59

## Confidential and Proprietary Property of Rocky Nevin

### Populate.java

25

```
(Name = path + separator + list[i]);
tFile = new File(tName);
if (tFile != null) {
    tFile_node = create_file_tree(tFile, current_depth+1);
    if (tFile_node != null) {
        mod = (this.file_mod - tFile.lastModified())/60000;// is it in millisec
        tFile_node.set_mag((Node.BARELY_VISIBLE_MAG);
        this.file_node.link(tFile_node, get_node_from_file_suffix(list[i]));
    }
}

onds? or even a valid time?

if (starting) {
    FilesNode.link(this.file_node);
    datasea.needdistUpdate = true;
}

return(this_file_node);
} // end create_file_tree

/**
 ** popn NOTES
 **
 */
public void popn () { // "show notes"
    int i;
    Node tn = datasea.find_node_named("Notes");
    if (tn==null) {
        GUI.ERRORROR(0, "", "node 'Notes' is null");
        return;
    }
    GUI.input.string_input("input Notes This is a free-form note");
    GUI.input.string_input("input Notes Tallis is red");
    GUI.input.string_input("input Notes Tallis is a cat");
    GUI.input.string_input("input Notes Tallis eats mice");
    GUI.input.string_input("input Carl called about computer");
    GUI.input.string_input("input Bob complained about desk");
    GUI.input.string_input("input John Smith called about car");

    GUI.input.string_input("input klavier:syn:piano");
    GUI.input.string_input("input Bob's Klavier is German");
    GUI.input.string_input("input Bob's piano is a Steinway");
    GUI.input.string_input("input Bob mlg 4pm NextWeek");
    GUI.input.string_input("input Bob mlg 4pm Oct 1999");
    GUI.input.string_input("input Mte with Jill 4pm Jun 3 2000");

    GUI.input.string_input("input Mlg with IBMer 4pm Apr 3 2000");
    GUI.input.string_input("input Bob is an IBMer");
    GUI.input.string_input("input Jill is an IBMer");

    //triple("Carl", "Red", "HairColor");
    //triple("Carl", "BMW", "Car");
    //GUI.input.string_input("input Carl's HairColor is Red");
    //GUI.input.string_input("input Carl's Car is BMW");

    GUI.input.string_input("input Mlg with Bob LastWeek");
    GUI.input.string_input("input Mlg with Bob Today");
    GUI.input.string_input("input Mlg with CommitteeX May 30 2000");

    datasea.needdistUpdate = true;
    GUI.PR0, "popn", "show Notes");
    return;
} // end (notes)

/**
 ** popx EXTRA, MISC
 **
 */
public void popx () { // "show web"
    int i;
    Node tn=null;

    GUI.PR0, "", "Populate miscellaneous, Web & Files Run.");
    gen_array(Cluster.Web);

    Cluster.Files.link(gen_tree(Cluster.Files));

    if ((tn = datasea.find_DN_named("D0.1.0")) != null)
        tn.link(datasea.cl("Music", "AN"));
    if ((tn = datasea.find_DN_named("D0.1.1")) != null)
        tn.link(datasea.cl("Music", "AN"));
    if ((tn = datasea.find_DN_named("D1.1.1")) != null)
        tn.link(datasea.cl("Music", "AN"));
    if ((tn = datasea.find_DN_named("D1.2.0")) != null)
        tn.link(datasea.cl("Theater", "AN"));
    if ((tn = datasea.find_DN_named("D1.2.1")) != null)
        tn.link(datasea.cl("Theater", "AN"));
    if ((tn = datasea.find_DN_named("D0.2.2")) != null)
        tn.link(datasea.cl("Theater", "AN"));
```

```

GUI.P(0, "popx", "show Web");
} // end popx

/**
 ** popmap
 **
 */
public void popmap () { // "show map"
    int i;
    Node map_node, SF_node, LA_node, Berkeley_node;
    // create a map

    map_node=new Node("Map","DN","Map of California", 100,100, 60,150);
    LA_node = create_node("LA","DN");
    SF_node = create_node("SF","DN");
    Berkeley_node = create_node("Berkeley","DN");

    LA_node.X = 10; LA_node.Y = -140;
    SF_node.X = 10; SF_node.Y = 120;
    Berkeley_node.X = 20; Berkeley_node.Y = 4;

    map_node.link(LA_node);
    LA_node.link(SF_node);
    Berkeley_node.link(SF_node);

    DataSea.Root.link(map_node);

    GUI.P(0, "popmap", "show Map");
} // end popmap

/**
 ** popfab
 **
 */
public void popfab () { // "show fab"
    int i;
    Node fab_node, X_node, Y_node, Z_node;
    Node node_1, node_2;
    // create a wafer fab data set with temp-machine data

    GUI.P(0, "ab", "late VR, e.g. 'Fab' Run.");
    fab_node=new Node("Fab","DN","Wafer Fab with machines and data", 100,100,600,400);
    X_node= new Node("Mach.X","DN","Machine X in wafer fab",0,-30,10,5);
    Y_node= new Node("Mach.Y","DN","Machine Y in wafer fab",50-30,10,5);

    Z_node= new Node("Mach.Z","DN","Machine Z in wafer fab",90,-30,10,5);
    fab_node.link(X_node);
    fab_node.link(Y_node);
    fab_node.link(Z_node);
    (fab_node.getLinkTo(X_node)).setLinks VRparams(X_node);
    (fab_node.getLinkTo(Y_node)).setLinks VRparams(Y_node);
    (fab_node.getLinkTo(Z_node)).setLinks VRparams(Z_node);

    node_1 = new Node("MachName","AN");
    node_1.link(X_node);
    node_1.link(Y_node);
    node_1.link(Z_node);

    node_1 = new Node("Temp","AN");
    node_2 = new Node("32","DN");
    node_1.link(node_2);
    X_node.link(node_2);
    node_2 = new Node("12","DN");
    node_1.link(node_2);
    node_1.link(node_2);
    Y_node.link(node_2);
    node_2 = new Node("21","DN");
    node_1.link(node_2);
    Z_node.link(node_2);

    node_1 = new Node("Voltage","AN");
    node_2 = new Node("v32","DN");
    node_1.link(node_2);
    X_node.link(node_2);
    node_2 = new Node("v12","DN");
    node_1.link(node_2);
    node_1.link(node_2);
    Y_node.link(node_2);
    node_2 = new Node("v21","DN");
    node_1.link(node_2);
    Z_node.link(node_2);

    GUI.P(0, "popfab", "show Fab");
    datasea.needsToUpdate = true;
} // end popfab (VR)

/**
 ** pope
 **
 */
public void pope () { // "show chromosome"

```

05/23/00  
00:16:59

# Confidential and Proprietary Property of Rocky Nevin

## Populate.java

27

```
int i;
Node a,b,c,d;
// create a chromosome with a few sites

a=new Node("Chromosome", "chromosome", "Fake chromosome");
DataSea.Root.link(a);

dataseta.needsToUpdate = true;
GUI.P(0, "popc", "show chromosome");
} // end (Chromosome)

/**
 ** DataSea.gen_array
 **
 */
public void gen_array (Node caller) {
    int i,j,k;
    Vector vec;
    Node tnode;
    int array_size = 20, x,y;

    Node CN3, CN4, CN5, CN6;

    vec = new Vector();
    for (i=0; i<array_size; i++) {
        x = (int)(caller.x +50-100*dataseta.gui.random());
        y = (int)(caller.y +50-100*dataseta.gui.random());
        tnode = new Node("el"+i, "DN", "from gen_array",
            x,y);
        vec.addElement(tnode);
        caller.link(tnode);
    }

    CN3 = new Node("FirstHalf", "AN");
    for (i=0; i<array_size/2; i++)
        CN3.link((Node)vec.elementAt(i));

    CN4 = new Node("SecondHalf", "AN");
    for (i=array_size/2; i<array_size; i++)
        CN4.link((Node)vec.elementAt(i));

    /**
     * CN5 = new Node("CN5", "DN");
     * for (i=0; i<array_size; i+=5)
     *     CN5.link((Node)vec.elementAt(i));
     *
     * CN6 = new Node("CN6", "DN");
     */

    for (i=0; i<array_size; i+=6)
        CN6.link((Node)vec.elementAt(i));
    caller.link(CN3);
    caller.link(CN4);
    caller.link(CN5);
    caller.link(CN6);
}

return;

/**
 ** gen_tree
 **
 */
public Node gen_tree (Node caller) {
    int i,j,k;
    int branch_count = 4;
    int x, y;

    Node trunk, node_i, node_j, node_k;

    //x = (int)caller.x;
    //y = (int)caller.y;
    x = 0;
    y = 0;

    trunk = new Node("A", "DN", "Trunk of Tree",
        0,0, 25,25);
    for (i=0; i<branch_count; i++) {
        node_j = new Node("B"+i, "DN", "branch_node",
            0, 0, 5, 5);
        trunk.link(node_j);

        for (j=0; j<branch_count; j++) {
            node_k = new Node("C"+i+"."+j, "DN", "branch_node",
                0,0, 2,2);
            node_j.link(node_k);

            for (k=0; k<branch_count; k++) {
                node_k = new Node("D"+i+"."+j+"."+k, "DN", "leaf node",
                    0,0, 2,2);
                node_k.link(node_k);
                if (k==0)
                    node_j.Type="DN";
            }
        }
    }
}
```

```
} // end gen_tree

return(trunk);
}

/**
 ** URLioANpair see if it exists, create it if need be, return it
 **
 */
public Node URLioANpair (String URL_name, String AN_name) {
    Node URL, AN;

//GUI.P(0,"URLioANpair", "Creating <"+URL_name+">, <"+AN_name+">");

    URL = create_node(URL_name, "URL", false);
    AN = create_node(AN_name, "AN", false);

    //URL.link(AN);
    URL.link(AN, Cluster_Web);

return(URL);
} // end URLioANpair

/**
 ** URLioURLpair see if it exists, create it if need be, return it
 **
 */
public Node URLioURLpair (String URL1_name, String URL2_name) {
    Node URL1, URL2;

//GUI.P(0,"URLioURLpair", "Creating <"+URL1_name+">, <"+URL2_name+">");

    URL1 = create_node(URL1_name, "URL", false);
    URL2 = create_node(URL2_name, "URL", false);

    //URL1.link(URL2);
    URL1.link(URL2, Cluster_Web);

return(URL1);
} // end URLioURLpair

/**
 ** ANpair see if it exists, create it if need be, return it
 **
 */
public Node ANpair (String AN1_name, String AN2_name) {
    Node AN1, AN2;

//GUI.P(0,"ANpair", "Creating <"+AN1_name+">, <"+AN2_name+">");

    AN1 = create_node(AN1_name, "AN", false);
    AN2 = create_node(AN2_name, "AN", false);

    AN1.link(AN2);

return(AN1);
} // end ANpair

/**
 ** METAPair see if it exists, create it if need be, return it
 **
 */
public Node METAPair (String meta_name, String AN_name, String URL_name) {
    Node meta, AN, URL;

//GUI.P(0,"METAPair", "Creating meta name of <"+meta_name+">, value of <"+AN_name+"> link
//ed to URL "+URL_name);

    meta = create_node(meta_name, "PN", false);
    AN = create_node(AN_name, "AN", false);
    URL = create_node(URL_name, "URL", false);

    meta.link(AN);
    URL.link(AN);
    //meta.isPolarized = true;
    //AN.isPolarized = true;

dataset.gui.dump_node(0, true, meta);

return(AN);
} // end METAPair
```

```
/**
 ** ANANpair see if it exists, create it if need be, return it
 **
 */
public Node ANANpair (String AN1_name, String AN2_name) {
    Node AN1, AN2;
    Node subnode;
    int num_words, i;
    String words[];

//GUI.P(0,"ANANpair","Creating <"+AN1_name+">,<"+AN2_name+">");

    AN1 = create_node(AN1_name, "AN", false);
    AN2 = create_node(AN2_name, "AN", false);

    AN1.link(AN2, "polarized");
    //AN1.isPolarized = true;
    //AN2.isPolarized = true;

// NOW SEE IF EITHER ARG CONSISTS OF INDIVIDUAL WORDS, MAKE UNPOLARIZE
D LINKS

    StringTokenizer t = new StringTokenizer(AN1_name, " ");
    num_words = t.countTokens();
    if (num_words != 1) { // if only one word, no need to break it apart
        words = new String[num_words];
        for(i = 0; i < num_words; i++) {
            words[i] = t.nextToken();
            subnode = create_node(words[i], "AN", false);
            AN1.link(subnode, "unpolarized"); // unpolarized link
            //GUI.P(0,"ANANpair","Linking AN1<"+AN1_name+">,<"+subnode.Name
            +">");
        }
    }

    /*****
    t = new StringTokenizer(AN2_name, " ");
    num_words = t.countTokens();
    if (num_words != 1) { // if only one word, no need to break it apart
        words = new String[num_words];
        for(i = 0; i < num_words; i++) {
            words[i] = t.nextToken();
            subnode = create_node(words[i], "AN", false);
            AN2.link(subnode, "unpolarized"); // unpolarized link
        }
    }
    *****/

    return(AN1);
} // end ANDNpair

return(AN1);
} // end ANDNpair

//GUI.P(0,"ANDNpair","Creating <"+AN1_name+">,<"+DN2_name+">");

    AN1 = create_node(AN1_name, "AN", false);
    DN2 = create_node(DN2_name, "DN", false);

    AN1.link(DN2);
    //AN1.isPolarized = true;
    //DN2.isPolarized = true;

    return(AN1);
} // end ANDNpair

/**
 ** triplet see if it exists, create it if need be, return it
 **
 ** Automatically create a matching AN for a DN
 */
public Node triplet (String DN1_name, String DN2_name, String AN_name) {
    Node DN1, DN2, AN;

    return(triplet(DN1_name, DN2_name, AN_name, null, null));
} // end triplet

/**
 ** triplet see if it exists, create it if need be, return it
 **
 ** Automatically create a matching AN for a DN
 */
public Node triplet (String DN1_name, String DN2_name, String AN_name,
    AN2.link(subnode, "unpolarized"); // unpolarized link
}
```

05/23/00  
00:16:59

## Confidential and Proprietary Property of Rocky Nevin

### Populate.java

30

```
Node DN1, DN2, AN;

Link link;

GUI.P(1, "triplet", "Creating <"+DN1_name+">, <"+DN2_name+">, <"+AN_name+">");

DN1 = create_node(DN1_name, "DN", false);
DN2 = create_node(DN2_name, "DN", false);
AN = create_node(AN_name, "AN", false);

// this way to polarize from DN1->DN2 and AN->DN2
DN1.link(DN2, CNode, polarized_string);
AN.link(DN2, CNode, polarized_string);

link = DN1.getLinkTo(DN2);
if (link != null)
    link.Name = AN.Name;

// AN.isCN = true;
// link.addCNode(AN); // have AN modulate link between DN1 and DN2
// link = DN2.getLinkTo(AN);
// link.addCNode(AN); // have AN modulate links to itself

// AN.isPolarized = true;
// DN1.isPolarized = true;
// DN2.isPolarized = true;
return(DN1);
} // end triplet

/**
 ** make_BB
 **
 */
public Node make_BB (String cmd, String Name) {
    int i, size;
    Node newBBnode;
    //String type = "BB";
    String type = "DN";

    if (cmd.equalsIgnoreCase("start")) {
        lastBBnode = create_node_forced(Name, type, "");
        GUI.P(0, "make_BB", "Created node "+lastBBnode.Name+" , Type="+lastBBnode.Type);
    }
    else if (cmd.equalsIgnoreCase("add")) {
        newBBnode = create_node_forced(Name, type, "");
        lastBBnode.link(newBBnode);
        GUI.P(0, "make_BB", "Created node "+newBBnode.Name+" , Type="+newBBnode.Type
            +", linked to "+lastBBnode.Name);
        lastBBnode = newBBnode; // save this one for later calls
    }
    return(lastBBnode);
} // end make_BB

/**
 ** create_node Name only, ForceCreation=false
 **
 */
public Node create_node (String Name) {
    return( create_node(Name, "DN", "", false) );
}

/**
 ** create_node Name and Type only, ForceCreation=false
 **
 */
public Node create_node (String Name, String Type) {
    return( create_node(Name, Type, "", false) );
}

/**
 ** create_node Name, Type and ForceCreation only
 **
 */
public Node create_node (String Name, String Type, boolean ForceCreation) {
    return( create_node(Name, Type, "", ForceCreation) );
}

/**
 ** create_node Name, Type and Desc only, ForceCreation=false
 **
 */
public Node create_node (String Name, String Type, String Desc) {
    return( create_node(Name, Type, Desc, false) );
} // end create_node

/**
 ** create_node Full version: see if it exists, create it if need be (or is forced).
 **
 */
```

05/23/00  
00:16:59

## Confidential and Proprietary Property of Rocky Nevin

### Populate.java

31

```

**
**      return it
**      (Automatically create a matching AN for a DN if >1 DN's of same name)
*/
public Node create_node (String Name, String Type, String Desc, boolean ForceCreation
) {
    Node ret_node=null, tnode=null;

    if (ForceCreation) {
        datasea.gui.P(0, "create_node", "ForceCreate is true!!!!!!!!!!!!!!!!!!!!");
        if (Type.equals("AN")) { // If forcing an AN, make top-level then a child
            tnode = datasea.find_node_named(Name, Type);
            if (tnode == null) { // get top-level AN, then add another AN
                tnode = create_node_forced(Name, Type, Desc);
            }
            ret_node = create_node_forced(Name, Type, Desc);
            tnode.link(ret_node);
            return(ret_node);
        }
        else {
            return(create_node_forced(Name, Type, Desc));
        }
    }
    else // don't ForceCreation
        if (((ret_node=datasea.find_node_named(Name))==null) {
            ret_node = new Node(Name, Type, Desc);
        }
        else {
            if (ret_node.Desc.equals(""))
                ret_node.Desc = Desc;
        }

    return(ret_node);
} // end create_node

/**
**      create_node_forced
**
*/
public Node create_node_forced (String Name, String Type) {
    return(create_node_forced(Name, Type, ""));
} // end create_node forced

/**
**
**      create_node_forced
**
*/
public Node create_node_forced (String Name, String Type, String Desc) {
    Node ret_node=null, other_dn_node=null, an_node=null;
    int i, size;

    if (Type.equalsIgnoreCase("DN")) {
        // see if there's an AN and link it to the new DN
        if there's a DN already, make sure an AN exists and link them
        if (null != (other_dn_node=datasea.find_node_named(Name, "DN"))) {
            an_node = datasea.find_node_named(Name, "AN");
            if (an_node == null) {
                an_node = create_node(Name, "AN", false);
                other_dn_node.link(an_node);
            }
        }
        ret_node = new Node(Name, Type, Desc); // forced
        ret_node.link(an_node); // may or may not exist
    }
    else
        ret_node = new Node(Name, Type, Desc); // forced

    return(ret_node);
} // end create_node_forced

/**
**      get
**
*/
public void get_a_URL (Node node) {
    int i, size=0;
    Node child;

    System.err.println("===== get_a_URL ... Begun ...=====");
    System.err.println("===== "+node.Name+", Type="+node.Type+"=====");
    if (node.isURL) {

```

05/23/00  
00:16:59

Confidential and Proprietary Property of Rocky Nevin

Populate.java

32

```
System.err.println("==== Calling pull_in_URLs on "+node.Name+"====");
pull_in_URLs(node.Name, "", 1, (Node)null);
}
else { // do it on all the children
    System.err.println("==== Calling pull_in_URLs on children of "+node.Name+"====");
    size = node.links.size();
    for (i=0; i<size; i++) {
        child = node.getNodeAtLink(i);
        if (child.isURL && (child.dist > node.dist))
            pull_in_URLs(child.Name, "", 1, (Node)null);
    }
}

} // end get_a_URL

} // end Populate
```

05/23/00  
00:34:55

GUI.java

```
// This is GUI.java by Rocky Nevin
import java.applet.*;
import java.lang.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.sql.*; // for the class Timestamp
import java.io.*;
```

```
/**
 * This is GUI.java by Rocky Nevin
 * which is the top-level object for DataSea, instantiated within this.
 */
```

```
LeftButton
MiddleButton -> Alt down
RightButton -> Meta down
```

```
*
* @version 0.1, 8/10/98
* @version 0.2, 9/6/98
* @version 0.3, 11/4/98
* run() -> update() -> [ paint(Node, layer_index) calls position_and_render_start(Node caller, int
dist.to_POV)]
* @version 0.4, 3/12/98
*
```

```
* Principal Methods:
add_buttons
demo1/2
dump
init
main
position_start/child_node/recursive
render_start/recursive
run_thread
sleep
update, paint
```

```
2/6/99
The bottom functions (render/position) return boolean whether to recurse,
assuming a VR function handles things in node-dependent ways.
```

Node.VRyes, via VRyes(Node), individually allows setting of VR mode  
render\_start()

```
l-> render_recursive() lastly calls render_recursive()
l-> render_node()
position_start()
l-> position_recursive() lastly calls position_recursive()
ll-> position_node()
```

NEED: to correct nsr.position,XXXXXX functions, e.g. DIR,DIR,ENTRY

```
2/25/99
Focus on Timeline, storing POV's and not recursing through them (spread, etc)
```

```
6/1/99
shift, meta etc are key events, handled in key_input(), and may also
be queried in processEvent (either newButton or newFrame) by
checking (InputEvent(event).isShift(), isMeta(), etc.
```

```
6/6/99
Remember:
width depends on node.size.x which is constant and in data coordinates.
in VRmode, if node.x depends on another node, its node.x=other.x(+-)other.size.x/2
magscale and WindowOffsets occur only once, in drawing on screen, i.e. using map() fn
// height = (int)((child.size.y));
That is, node.x\X\dx\dx is in data space and should thus not depend on viewwin
g_parms
```

```
6/7/99
Very odd: I don't see why the old style of POV or Parent pressure
brings in remote high-value mags, e.g. from 'sim', but it does.
Probably the force is simply strongly negative, the direction tending
vertically from POV to its linked node.
Note also that 2pm:today: is not linked to Today
Note also 'text' for EMF is not placed nicely
Note large data text isn't handled.
```

```
6/8/99
Fixed the thread interaction problem (I think), whereby unlinking nodes while a
thread was animating, and insize a recursive rendering routine, would give invalid
references to pointers.
Changed child positioning to be fan-shaped, POV pressure now segregates higher mag
dial children.
Application EMail now automatically resets view and POV and zooms on the email form.
Looks good.
```

6/11/99

05/23/00  
00:34:55

# Confidential and Proprietary Property of Rocky Nevin GUI.java

2

Consider way to have parsed input linked distally to ANs, like email, phone, etc.

6/12/99

Problem with TL position with getlinksVRparams() is that I've not called setlinksVRparams() on all of them, therefore VRyes() returns false often.

6/17/99

Added vote branch, to vote on branches

Need to add time-dependent display, e.g. pulsating mags to see effects, 'what if's

6/18/99

Added draw\_string() which wraps text, and started popecon().

Showing TL, then resetting and showing again, it's children Today, Tomorrow, Yesterday are in different positions, same X though, same linkage as before.

6/21/99

Need abs to be more general, e.g. spreads through anything, not just DNS  
TL and friends is a mess, not positioned as expected.

6/27/99

Inhibited recursion if dist==1, added Node.importance

## END OF COMMENTS

\*/

public class GUI extends Applet implements Runnable {

int INFINITE\_DEPTH = 1000; // just a number to give essentially infinite depth of levels  
int image\_counter = 1;

public static int counter\_of\_positioned\_nodes = 0;

public static int max\_transition\_count = 1;

public static Graphics graphics;

public static Graphics graphics1, graphics2;

public static Graphics text\_graphics;

public static Frame frame, frame2, text\_frame, diag\_frame;

Image image1, image2;

static Mode mode\_obj;

static String GlobalUserDir;

static String GlobalUserHomeDir;

static String GlobalOSName;

static boolean quick = true;

static boolean doNormalization = true;

static boolean showBoundaries = false;

static boolean checkPolarization = true;

static boolean auto\_rescale = true;

static boolean parse = true;

static double spread\_factor=0.15;

static double thetaOffset=0.4;

static double max\_x, min\_x, max\_y, min\_y; // used by rescale() and auto\_rescale

static int desired\_visible\_count = 30; // used by DataSea.auto\_flatten() 8/24/99

Point GlobalMapPoint; //

static Point TinyPoint; // used by ma ()

Force force;

int drawing\_counter=0;

static boolean Animating = false;

static boolean NetOK = true;

static boolean shrinking\_allowed = true;

static boolean is\_meta\_down = false;

static boolean is\_alt\_down = false;

static boolean is\_control\_down = false;

static boolean is\_shift\_down = false;

static boolean coordinates\_on=true;

static boolean drawBoxes = true;

static boolean TinyFlag = false;

static boolean globalDoTiny = false;

static boolean global\_ok\_to\_draw = false;

static double global\_angle\_array[];

//static double TinyScale = 1.0;

static double globalMaxPressure = 1.0;

static boolean StopThreadRequest = false;

static boolean FlipAxes = false;

static boolean meta=false;

static boolean alt=false;

static boolean shift=false;

static boolean control=false;

static int counter=0;

static int mouseX=0;

static int mouseY=0;

static int spaceX=0;

static int spaceY=0;

static double magscale = 1.0;

double pressure\_mag = 3.0;

static int updateCount = 8;

static int Globaldist;

static double DEFAULT\_TEXT\_THRESHOLD=Node.BIG\_MAG;

static double DEFAULT\_POS\_THRESHOLD=Node.BARELY\_INVISIBLE\_MAG;

static double DEFAULT\_RELATIONS\_THRESHOLD=Node.MED\_MAG;

static double DEFAULT\_BOX\_THRESHOLD=Node.MED\_MAG;

static double pos\_threshold = Node.BARELY\_INVISIBLE\_MAG;

static double relations\_threshold = Node.MED\_MAG;

static double text\_threshold = Node.MED\_MAG-1;

static double box\_threshold = Node.MED\_MAG;

static Thread animation\_thread;

static GUI gui;

static DataSea dataset;

static Input input;

static Timer timer;

05/23/00  
00:34:55

## Confidential and Proprietary Property of Rocky Nevin

### GUI.java

3

```
static TL tl;
static int recursion_depth = 1;
static int dist_limit = 100;
static boolean drawText = true;
static boolean drawLinkNames = true;
static boolean drawFile = false;
static boolean drawDirectory = true;
static boolean drawWind = true;
static boolean drawAN = true;
static boolean drawON = true;
static boolean drawDN = true;
static boolean drawEvent = true;
static boolean drawCN = false;
static boolean drawPN = true;
static boolean drawURL = true;
static Node Xnode=null;
static Node lastNode=null;
static Node TinyNode = null;
static Node SavedNode = null;
static Node Myself; // This is a self-node, one which can be used to show the DataSea
// program itself
static Node VR_MACH_NODE, VR_FAB_NODE;
boolean string_active = false;
String TileString = "";
String key_input_string = "";
static String lastCommand = "";
static String StatusLine[];
static String global_str[] = null;
static int global_str_size=0;
static int MAX_GLOBAL_STR_SIZE=140;
Timestamp timestamp;
java.util.Date date;
static java.lang.Double java_lang_double;
static java.lang.Long java_lang_long;
// Node node;
static c_long current_TS;
static c_long thisCommandTS;
static c_long lastCommandTS;
static int Debug = 0;
static boolean Details = true;
static int GlobalNodeNode = 0;
static Font titleFont;
static Font littleFont;
Font buttonFont;
// static int WindowWidth=600, WindowHeight=600;
static int WindowWidth=1240, WindowHeight=1028; // defaults, set in init()
```

---

```
static int WindowXcenter=WindowWidth/2, WindowYcenter=WindowHeight/2;
static int WindowXOffset=0, WindowYOffset=0;
static String priorCommand = "";
static Vector selected_nodes_vec; // holds all nodes that are selected
public static KeyEvent ke;
TextField text_field;

static List list;
MyDialog query_dialog, input_dialog;
static newButton button_animate;
static newButton button_reset;
static newButton button_absorb_POV;
static newButton button_populate;
static newButton button_position;
static newButton button_background_color;
static newButton button_render;
static newButton button_lines;
static newButton button_potential;
static newButton button_postprocessor;
static newButton button_increase_mag;
static newButton button_Debug;
static newButton button_Dump;
static newButton button_more_attraction;
static newButton button_more_repulsion;
static newButton button_Stop;
static newButton button_drawText;
static newButton button_drawDN;
static newButton button_drawCN;
static newButton button_drawParent;
static newButton button_presPOV;
static newButton button_presNeighbors;
static newButton button_presNoise;
static Checkbox check;
static double ThetaMultiplier=Math.PI/2;

ColorObj color_obj;

public static void main (String[] argv)
{
    gui=new GUI();
}

public GUI() {
    super();
```

05/23/00  
00:34:55

## Confidential and Proprietary Property of Rocky Nevin

### GUI.java

4

```
Graphics[] graphicsarray;
if (gui == null)
    init();
}

/**
 * beep
 */
static public void beep () {
    int i, size;
    Note tn;

    Toolkit.getDefaultToolkit().beep();

    } // end beep

/**
 * If G is the top level object of an applet, init() gets run automatically.
 * Linked objects and animation
 */
public void init () { // frame, graphics, font
    int i;

    if (gui == null)
        gui = this;

    dump_properties();

    global_sfr = new String[MAX_GLOBAL_STR_SIZE];
    StatusLine = new String[15];
    StatusLine[0] = "-----";
    for (i=0; i<15; i++)
        StatusLine[i] = "";

    Toolkit.getDefaultToolkit().beep();
    System.out.println(Toolkit.getDefaultToolkit().getScreenSize());
    Window.Width=Toolkit.getDefaultToolkit().getScreenSize().width;
    Window.Height=Toolkit.getDefaultToolkit().getScreenSize().height;
    Window.Xcenter=Window.Width/2;
    Window.Ycenter=3*Window.Height/5;

    timer = new Timer();
    inout = new Inout(gui);

    color_obj = new ColorObj();
    color_obj.init();
    reset_current_TS0;
    date = new java.util.Date();
    timestamp = new Timestamp(date.getTime());
    frame = new JFrame("This is a Frame");
    frame.setSize( Window.Width, Window.Height);
    Point p = new Point(20,0);
    frame.setLocation( p );
    // frame.setBackground(Color.white);
    frame.setBackground(color_obj.DarkGrey);
    //frame.setBackground(color_obj.VeryLightGrey);
    frame.setLayout(new java.awt.FlowLayout()); // 8/22/99
    //frame.setLayout(new java.awt.BorderLayout());
    frame.setTitle("DataSea: "+date.toString());
    titleString = "DATABASE: CONFIDENTIAL AND PROPRIETARY Information of Rocky Nevin
    , "+date.toString();
    frame.setTitle(titleString);
    //frame.show();
    //frame.setVisible(true);
    graphics = frame.getGraphics();
    status("Hello, this is GUI.init() running. OS_Name="+GlobalOSName+", UserID="+GlobalUser
    Dir);
    titleFont = new java.awt.Font("Arial", Font.BOLD, 12);
    titleFont = new java.awt.Font("Helvetica", Font.BOLD, 8);
    frame.setFont(titleFont);
    frame.setVisible(true);

    text_frame = new Frame("URL Frame");
    text_frame.setSize( 200, 400 );
    p = new Point(Window.Width-200, 0);
    text_frame.setLocation( p );
    text_frame.setBackground(color_obj.VeryLightBlue);
    //text_frame.setLayout(new java.awt.BorderLayout());
    // TitleString =
    //text_frame.setTitle(titleString);
    //text_frame.show();
    //text_graphics = text_frame.getGraphics();

    diag_frame = new Frame("Diagnostic Messages");
    diag_frame.setSize( Window.Width, 150 );
    p = new Point(0, Window.Height - 130);
    diag_frame.setLocation( p );
    diag_frame.setBackground(Color.black);
```

05/23/00  
00:34:55

## Confidential and Proprietary Property of Rocky Nevin GUI.java

5

```
diag.frame.setBackground(color.obj.VeryLightGreen);
list = new List(22);
list.setEnabled(true);
diag.frame.add("North", list);

mode.obj = new Mode();
force = new Force();
GlobalMapPoint = new Point();
java.lang.double = new Double(0);
java.lang.Long = new Long(0);

add_buttons():
    image1 = frame.createImage(WindowWidth, WindowHeight);
    image2 = frame.createImage(WindowWidth, WindowHeight);
    graphics1 = image1.getGraphics();
    graphics2 = image2.getGraphics();

    datasca = new DataSea(this);
    input = new Input(this);
    tl = new TL(); // Timeline object

    // Construct Data Sea
    // Construct Data Sea

    datasca.pop.populate_begin();

    // test();

    run_thread();

    data:sea.word_reset();
    global_ok_to_draw = true;

    } // end init

/**
 ** write_to_frame2
 */
public void write_to_frame2 (String s) {

    graphics.frame2.clearRect(0,0, (frame2.getSize().width, (frame2.getSize().height);

    date = new java.util.Date();
    TitleString = "DATASEA: CONFIDENTIAL AND PROPRIETARY Information of Rocky Nevin
    "+date.toString();

    graphics.frame2.setColor(Color.blue);
    graphics.frame2.drawString(date.toString(), 10,40);

    graphics.frame2.setColor(Color.black);
    graphics.frame2.drawString("abcdefg counter = "+counter, 10,100);
    graphics.frame2.drawString(s, 10,110);

    return;
    } // end write_to_frame2

/**
 ** extra_frame
 **
 */
    public void extra_frame () {
        frame2 = new JFrame("This is a Frame");
        frame2.setSize(600, 400);
        frame2.setLocation( new Point(0,(int)(WindowHeight*0.8)) );
        frame2.setBackground(color.obj.VeryLightGrey);
        frame2.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame2.setTitle("Frame2: "+date.toString());
        frame2.setVisible(true);
        graphics.frame2 = frame2.getGraphics();
        sleep(1000); // need to wait before writing to it
    } // end extra_frame

    public void test () {
        int c;
        char b[];
        b = new char[1000];

        extra_frame();
        write_to_frame2("Hello, this is a test from extra_frame()");
    } // try {

        File inputFile = new File("usr/people/rocky");
        // FileInputStream fis = new FileInputStream(inputFile);
        // while ( (c=fis.read(b)) != -1)
        //     System.err.print("!:>>>"+b[0]+b[1]+b[2]+b[3]+b[4]+b[5]+"<<<");
        // fis.close();
        //
        // FileReader fr = new FileReader(inputFile);
```

```

file_dump(inputFile);
// }

// catch (FileNotFoundException e) { ERROR(0,"test","File /usr/people/rocky/x not fou
nd."); }

// catch (IOException e) { ERROR(0,"test","IOException."); }

calc_angles();

/*****
 * FontMetrics fontMetrics;
 * fontMetrics = new FontMetrics(titleFont);
 * System.err.println("FontMetrics.charsWidth()="
 * +fontMetrics.charsWidth(chars,0,10));
 *****/

/*****
 * P(0,"init","theta(1,0)=" +get_angle(1,0));
 * P(0,"init","theta(1,1)=" +get_angle(1,1));
 * P(0,"init","theta(0,1)=" +get_angle(0,1));
 * P(0,"init","theta(-1,0)=" +get_angle(-1,0));
 * P(0,"init","theta(-1,-1)=" +get_angle(-1,-1));
 * P(0,"init","theta(0,-1)=" +get_angle(0,-1));
 *****/

/*****
 * R runtime = Runtime.getRuntime();
 * System.err.println("init(): Runtime.freeMemory() is "+rt.freeMemory());
 * try { rt.exec("date"); }
 * catch (java.io.IOException e) { System.err.println("Runtime IOException: "+e.toString()); }
 *****/

} // end test

/**
 ** file_dump
 **
 */

public void file_dump (File inputFile) {
int i;
String list[], tName;
File tFile;
char separator;

separator = inputFile.separatorChar;
long mod;

P(0,"file_dump", "pathSeparator="+inputFile.pathSeparator);
P(0,"file_dump", "separator="+inputFile.separator);
P(0,"file_dump", "Name="+inputFile.getName());
P(0,"file_dump", "Parent="+inputFile.getParent());
P(0,"file_dump", "isDirectory="+inputFile.isDirectory());
list = inputFile.list();
P(0,"file_dump", ""+list.length+" entries ...");
for (i=0; i<list.length; i++) {
tName = inputFile.getPath() + separator + list[i];
mod = tFile.lastModified();
mod /= 60000;
if (tFile.isDirectory())
P(0,"file_dump", i+"Dir: "+tName+" , last modified "+mod+" min's ago");
else
P(0,"file_dump", i+"File: "+tName+" , last modified "+mod+" min's ago");
}

return;
} // end file_dump

/**
 ** calc_angles
 **
 */

static public void calc_angles () {
double theta=0;
int i;
double x, y;
long TS1=0, TS2=0;

timer.start_timer("nothing");
timer.end_timer("nothing");

timer.start_timer("allocating 10000 doubles");
global_angle_array = new double[10000];
timer.end_timer("allocating 10000 doubles");

TS1 = java.lang.System.currentTimeMillis();
TS2 = java.lang.System.currentTimeMillis();
System.err.println("Time of nothing is "+(TS2-TS1)+" milliseconds");

```

05/23/00  
00:34:55

GUI.java

7

```

timer.start(timer("get angles");
TS1 = java.lang.System.currentTimeMillis();
for (i=0; i<10000; i++) {
    theta = get_angle((double)i,(double)1.3);
}
TS2 = java.lang.System.currentTimeMillis();
System.err.println("Time of 10000 get_angle's is "+(TS2-TS1)+" milliseconds");
timer.end(timer("get angles");

TS1 = java.lang.System.currentTimeMillis();
for (i=0; i<10000; i++) {
    theta = Math.atan2((double)i,(double)1.3);
}
TS2 = java.lang.System.currentTimeMillis();
System.err.println("Time of 10000 atan2's is "+(TS2-TS1)+" milliseconds");

TS1 = java.lang.System.currentTimeMillis();
for (i=0; i<10000; i++) {
    theta = i * (2*Math.PI / 10000.0);
    global_angle_array[i] = Math.sin(theta);
}
TS2 = java.lang.System.currentTimeMillis();
System.err.println("Time of 10000 sin(x) calc's into array is "+(TS2-TS1)+" milliseconds");

// Time doing a thousand sine's
TS1 = java.lang.System.currentTimeMillis();
for (i=0; i<10000; i++) {
    theta = i * (2*Math.PI / 10000.0);
    x = Math.sin(theta);
    if (TS1>TS2)
        y = x;
}
TS2 = java.lang.System.currentTimeMillis();
System.err.println("Time of 10000 sin(x) is "+(TS2-TS1)+" milliseconds");

// Time retrieving a thousand elements
TS1 = java.lang.System.currentTimeMillis();
System.err.println("Time of printing above "+(TS1-TS2)+" milliseconds");
TS1 = java.lang.System.currentTimeMillis();
for (i=0; i<10000; i++) {
    theta = i * (2*Math.PI / 10000.0);
    x = global_angle_array[i];
    if (TS1>TS2)
        v = x;
}

TS2 = java.lang.System.currentTimeMillis();
System.err.println("Time of 10000 elements of global_angle_array is "+(TS2-TS1)+" milliseconds");
");

}

TS2 = java.lang.System.currentTimeMillis();
System.err.println("Time of 10000 elements of global_angle_array is "+(TS2-TS1)+" milliseconds");

//**
//** dump.properties
//**
//**
public void dump_properties () {
    GlobalUserDir = System.getProperty("user.dir");
    GlobalUserHomeDir = System.getProperty("user.home");
    GlobalOSName = System.getProperty("os.name");

    System.err.println(" ");
    System.err.println("=====");
    ==");
    System.err.println("version:
    System.err.println("vendor:
    System.err.println("vendor.url:
    System.err.println("class.version:
    System.err.println("os.name:
    System.err.println("os.arch:
    System.err.println("file.separator:
    System.err.println("user.name:
    System.err.println("user.home:
    System.err.println("user.dir:

    System.err.println("class.path:
    System.err.println("path.separator:
    System.err.println("line.separator:

    System.err.println("=====");
    ==");
    System.err.println(" ");
    return:

```

```
// end dump.properties

/**
 ** clear
 */
public void clear () {
    int x1, y1, x2, y2;

    graphics.clearRect(0,0, (frame.getSize().width, (frame.getSize().height);
    if (coordinates.on) {
        graphics.setXORMode(colorObj.LightRed);
        map(-100, 0, GlobalMapPoint); x1 = GlobalMapPoint.x; y1 = GlobalMapPoint.y;
        map(100, 0, GlobalMapPoint); x2 = GlobalMapPoint.x; y2 = GlobalMapPoint.y;
        graphics.drawLine(x1, y1, x2, y2);

        map(0, 100, GlobalMapPoint); x1 = GlobalMapPoint.x; y1 = GlobalMapPoint.y;
        map(0, -100, GlobalMapPoint); x2 = GlobalMapPoint.x; y2 = GlobalMapPoint.y;
        graphics.drawLine(x1, y1, x2, y2);
        graphics.setPaintMode();
    }
} // end clear

/**
 ** amplify_region
 */
public void amplify_region(int val) {
    P(0, "amplify_region", "Got val="+val+", no action coded yet.");
} // end amplify_region

/*
 * key_input
 * this handles key input, if key is Enter, then accumulated
 * string 'GUI_key_input_string' is given to input.string.input()
 */
public void key_input (KeyEvent ke) {
    int key = ke.getKeyCode();
    String keystring = ke.getKeyText(key);
    char ch = ke.getKeyChar();

    P(2, "key_input", "Keystring is <"+keystring+">, KeyChar is <"+ch+">");

    if (keystring.equals("Shift")) {
    }
    else if (keystring.equals("Alt")) {
    }
    else if (keystring.equals("Ctrl")) {
    }
    else
    {
        if (keystring.equals("Enter")) {
            input.string.input(key_input_string);
            if (!key_input_string.equals("")) {
                lastCommand = key_input_string;
                p("lastCommand: "+lastCommand);
            }
        }
        key_input_string = "",
        status(lastCommand);
    }
} // HANDLE PLUS KEY and MINUS KEY plus key minus key
else if ((ch == '+') && (key_input_string.equals("")) && (lastNode != null)) {
    lastNode.more_mag();
    datasea.normalize();
}
else if ((ch == '-') && (key_input_string.equals("")) && (lastNode != null)) {
    dump_node(0, false, lastNode);
    lastNode.less_mag();
    datasea.normalize();
    dump_node(0, false, lastNode);
}
else if (keystring.equals("Up") || keystring.equals("Down")
|| keystring.equals("Left") || keystring.equals("Right"))
    arrow_key(keystring);
else if (keystring.equals("F1") || keystring.equals("F2")
|| keystring.equals("F3") || keystring.equals("F4")
|| keystring.equals("F5") || keystring.equals("F6"))
    Fkey(keystring);
else if (keystring.equals("Backspace")) {
    if (key_input_string.equals(""))
    ;
    else {
        key_input_string = key_input_string.substring(0, -1+key_input_string.length());
    }
}
else {
    key_input_string = key_input_string+ch;
}
}
```

05/23/00  
00:34:55

## Confidential and Proprietary Property of Rocky Nevin

GUI.java

9

```
) // end key_input

/**
 ** F_key(String)
 */
public void F_key (String str) {

    if (str.equals("F1")) {
        P(0,"F_key","F1: Help:");
        help();
    }

    if (str.equals("F2")) {
        if (lastNode == null)
            WARNING(0,"F_key", "F2: need a selected 'lastNode' to link to POV");
        else {
            P(0,"F_key","F2: linking POV to "+lastNode.Name);
            input_string_input("show "+lastNode.Name);
        }
    }

    if (str.equals("F3")) {
        if (lastNode==null)
            WARNING(0,"F_key", "F3: need a selected 'lastNode' to find common nodes");
        else {
            P(0,"F_key","F3: stimulate similar nodes to "+lastNode.Name);
            datasea.find_common_nodes.to(lastNode);
        }
    }

    if (str.equals("F4")) {
        P(0,"F_key","F4: change POV");
        if (lastNode != null)
            datasea.POV = lastNode;
    }

    if (str.equals("F5")) {
        P(0,"F_key","F5: unlink all and link lastNode to POV");
        solo_link_to_POV();
    }

    if (str.equals("F6")) {
        P(0,"F_key","F6: unlink lastNode");
        if (datasea.POV!=null)
            datasea.POV.unlink_both(lastNode);
        else
            WARNING(0,"word_selector","Need a POV to work.");
    }
}

return;
} // end F_key

/**
 ** solo_link_to_POV
 **
 */
public void solo_link_to_POV() {
    int i;
    Node a, b, c, d;
    if (datasea.POV == null) {
        WARNING(0,"solo_link_to_POV","Need a POV to work.");
        if (lastNode != null)
            datasea.show(lastNode);
        return;
    }
    if (lastNode == null) {
        WARNING(0,"solo_link_to_POV","Need a lastNode to work.");
        return;
    }
    stop.thread();
    datasea.POV.unlink_all();
    datasea.POV.link(lastNode);
    datasea.neeDistUpdate = true;
    //if (animation_thread == null) {
        // run_thread();
    // }
} // end solo_link_to_POV

/**
 **
 ** help
 */
public void help () {
    P(0,"help","n toggles NeighborPressure, l toggles lines");
    P(0,"help","abs [] magnifies abstractions of a DN, 'back []' mag's backwards to POV, 'most'
    =create POV and link to greatest mags");
    P(0,"help","ren arg1 [] =rename, 'link arg1 []', 'del []'=delete node");
    P(0,"help","rtv chooses rendering relations or VR, 'l' toggles lines, 'n' toggles neighbor pre
    ssure 'select', 'unselect (all)");
    P(0,"help","focus [] =make a node the POV, 'le', 'ri', 'up', 'do' =left, right, up, down; 'lup',
    'tdown' = text threshold up or down by 0.1");
    P(0,"help","popd, TL,f,m,n,s,c,x,VR, daldd=draw CNIDNs");
    P(0,"help","F4=turn lastNode into POV F5=unlink all from POV and link lastNode to POV F6
```

```

=unlink lastNode from POV");
P(0,"help","F1=help F2=connect lastNode to POV F3=show similar nodes to lastNode");
return;
}

/**
 ** shift_one_node
 **
 */
public void shift_one_node (Node node, String direction) {
    int i, size, dist_diff=1;
    Node child, saved_node=null;
    double saved_mag;

    if (node == null)
        return;

    saved_mag = 0;
    if (direction.equalsIgnoreCase("distal"))
        dist_diff = 1;
    if (direction.equalsIgnoreCase("proximal"))
        dist_diff = -1;

    size = node.links.size();
    for (i=0; i<size; i++) {
        child = node.getChild(i);
        if ((child.dist == (node.dist+dist_diff) && node!=dataBase.Root) { // check the direction
            if (child.mag > saved_mag) { // save the biggest candidate
                saved_mag = child.mag;
                saved_node = child;
            }
        }
    }

    if (saved_node != null) {
        P(0,"shift_one_node","direction "+direction+" ", "lastNode.Name+" -> "+saved_node.
        Name);
        lastNode = saved_node;
        if (lastNode.mag < Node.MED_MAG)
            lastNode.mag = Node.MED_MAG;
    }
}

} // end shift_one_node

/**
 ** arrow_key(String)
 **
 */
public void arrow_key (String str) {
    if (lastNode != null) {
        if (str.equals("down")) {
            if (str.equals("Up"))
                lastNode.Y += 10;
            if (str.equals("Down"))
                lastNode.Y -= 10;
            if (str.equals("Right"))
                lastNode.X += 10;
            if (str.equals("Left"))
                lastNode.X -= 10;
            P(0,"arrow_key","->For lastNode.Name="+lastNode.Name+" got "+str+" , changing no
            de.X/Y");
        }
        else {
            if (str.equals("Up"))
                shift_one_node(lastNode, "distal");
            if (str.equals("Down"))
                shift_one_node(lastNode, "proximal");
            if (str.equals("Right"))
                ;
            if (str.equals("Left"))
                ;
        }
    }
    else { // no lastNode available
        if (Debug == 2)
            P(2,"arrow_key","->Got "+str+" NO lastNode, shifting entire screen.");
        Window.YOffset += 100/magscale;
        if (str.equals("Down"))
            Window.YOffset -= 100/magscale;
        if (str.equals("Right"))
            Window.XOffset += 100/magscale;
        if (str.equals("Left"))
            Window.XOffset -= 100/magscale;
    }
}

```

```

    )

    /**
     ** handle mouse clicked
     ** @params Node node, the node already determined to be under the mouse
     ** Find the nearest node to the cursor.
     */
    public void handle_mouse_clicked (Node node) {

        lastNode = node;
        P i: "handle_mouse_clicked", "is meta_down="+is_meta_down+", is alt_down="+is_alt_down+", is c
        ontrol_down="+is_control_down+", is shift_down="+is_shift_down";

        // control keys are down ...
        if (GUI.control)
            solo_link_to_POV();
        else if (GUI.alt)
            action(node);
        else if (GUI.shift)
            dataseta.vote_branch(null, "+");
        else if (is meta_down) // Right mouse button
            dataseta.inhibit(node);
        else { // Left mouse button
            dump_node(0, false, node);
            ; // just set the node as lastNode
        }

        /*****
        action (alt)
        POV (ctl)
        vote - (meta ... Rmouse)
        vote + (shift)
        select (plain)
        *****/

        dat: sea.normalize();

        return;
    } // end handle_mouse_clicked

    /**
     ** mouse_clicked_at
     ** @params int x,y, the cursor position where the mouse is.
     ** Find the nearest node to the cursor.
    */
    public Node mouse_clicked_at (int x, int y) {
        Node node=null, tnode=null;
        double tdist, saved_dist = 100000000;
        int i, size;

        // Search through all nodes and find nearest one to where mouse was clicked
        size = dataseta.node_vec.size();
        for (i=0; i<size; i++) {
            tnode = (Node)dataseta.node_vec.elementAt(i);
            if (
                (tnode.mag >= Node.SMALL_MAG)
                && (!tnode.isURL || drawURL)
                && (!tnode.isAN || drawAN)
                && (!tnode.isFile || drawFile)
                && (!tnode.isDN || drawDN))) {
                tdist = (tnode.x-x)*(tnode.x-x) + (tnode.y+tnode.size.Y/2-y)*(tnode.y+tnode
                .size.Y/2-y);
                if (tdist < saved_dist) {
                    saved_dist = tdist;
                    node = tnode;
                }
            }
        }

        lastCommand = "Mouse selected '"+node.Name+"', mag="+node.mag+", d="+node.dist+", Child
        Cnt="+node.ChildCount;
        handle_mouse_clicked(node);
        return (node);
    } // end mouse_clicked_at

    /**
     ** newFrame instead of Frame
     */
    class newFrame extends Frame {
        newFrame(String name) {
            super(name);
            enableEvents(
                //
                // AWTEvent.ADJUSTMENT_EVENT_MASK
                //
                // AWTEvent.COMPONENT_EVENT_MASK
                //
                // AWTEvent.CONTAINER_EVENT_MASK
                //
                // AWTEvent.FOCUS_EVENT_MASK
                //
                // AWTEvent.ITEM_EVENT_MASK
                //
                // AWTEvent.KEY_EVENT_MASK
                //
                // AWTEvent.MOUSE_EVENT_MASK
                //
                // AWTEvent.MOUSE_MOTION_EVENT_MASK
                //
                // AWTEvent.TEXT_EVENT_MASK
            )
        }
    }

```

05/23/00  
00:34:55

## Confidential and Proprietary Property of Rocky Nevin

GUI.java

112

```
// IAWTEvent.WINDOW_EVENT_MASK
);
// Odd, I can't get KEY_PRESSED events from newFrame's processEvent, even with KEY_EVENT_MASK on

public void processEvent (AWTEvent event) { // newFrame
    int x, y;
    Node tn;

    if (event.getID() == MouseEvent.MOUSE_MOVED) {
        // ALL THIS IS TO GET COORDINATES OF MOUSE
        x = ((MouseEvent)event).getX();
        y = ((MouseEvent)event).getY();
        GUI.mouseX=x;
        GUI.mouseY=y;
        x -= WindowXcenter;
        y -= WindowYcenter;
        x /= magScale;
        y /= magScale;
        x += WindowXOffset;
        y -= WindowYOffset;
        GUI.spaceX=x;
        GUI.spaceY=-y;
        if (Debug == 4)
            P(4, "processEvent.newFrame",
                "(newFrame)event="+event.paramString()+
                ", X="+((MouseEvent)event).getX()+
                ", id="+event.getID());
    }
}
```

```
// How to determine if it's a MouseEvent or WindowEvent?
// Resizing the window apparently sends resize message to a button,
// and the button becomes the size of the window.
/*****
** if (event.getID() != MouseEvent.MOUSE_MOVED) {
**     P(0, "processEvent.newFrame", "toString()="+event.toString());
**     P(0, "processEvent.newFrame", "paramString()="+event.paramString());
**     P(0, "processEvent.newFrame", "getID()="+event.getID());
** }
** if (
**     (event.getID() != WindowEvent.WINDOW_ACTIVATED) &&
**     (event.getID() != WindowEvent.WINDOW_DEACTIVATED) &&
**     (event.getID() != WindowEvent.WINDOW_CLOSING) &&
**     (event.getID() != WindowEvent.WINDOW_CLOSED) &&

```

```
** (event.getID() != WindowEvent.WINDOW_OPENED) &&
** (event.getID() != WindowEvent.WINDOW_CLOSED) &&
** (event.getID() != WindowEvent.WINDOW_ICONIFIED) &&
** (event.getID() != WindowEvent.WINDOW_DEICONIFIED)
** )
*****/
{ // THIS IS NEVER RUN ...
    is_meta_down = ((InputEvent)event).isMetaDown();
    is_alt_down = ((InputEvent)event).isAltDown();
    is_control_down = ((InputEvent)event).isControlDown();
    if (is_control_down)
        System.out.println("----- is_control_down -----");
    is_shift_down = ((InputEvent)event).isShiftDown();
    GUI.meta = is_meta_down;
    GUI.alt = is_alt_down;
    GUI.shift = is_shift_down;
    GUI.control = is_control_down;
    if (is_control_down)
        P(0, "newFrame.processEvent", "Control down!!!");
    if (Debug==4) {
        if (is_meta_down)
            P(0, "newFrame.processEvent", "Meta down!!!");
        if (is_alt_down)
            P(0, "newFrame.processEvent", "Alt down!!!");
        if (is_control_down)
            P(0, "newFrame.processEvent", "Control down!!!");
        if (is_shift_down)
            P(0, "newFrame.processEvent", "Shift down!!!");
    }
}
```

```
// -----
// if (event.getID() == MouseEvent.MOUSE_CLICKED)
// {
//     tn=mouse.clicked.at(GUI.spaceX, GUI.spaceY); // I draw Y increasing going up
// } // End method processEvent()
// // End class newFrame

class newButton extends Button {
    String original_name;

    newButton(String name) {
        super(name);
        original_name = name; // Set original name
    }
}
```

05/23/00  
00:34:55

# Confidential and Proprietary Property of Rocky Nevin

GUI.java

13

```
enableEvents(
    AWTEvent.ACTION_EVENT_MASK
    | AWTEvent.ADJUSTMENT_EVENT_MASK
    | AWTEvent.COMPONENT_EVENT_MASK
    | AWTEvent.CONTAINER_EVENT_MASK
    | AWTEvent.FOCUS_EVENT_MASK
    | AWTEvent.ITEM_EVENT_MASK
    | AWTEvent.KEY_EVENT_MASK
    | AWTEvent.MOUSE_EVENT_MASK
    | AWTEvent.MOUSE_MOTION_EVENT_MASK
    | AWTEvent.TEXT_EVENT_MASK
    | AWTEvent.WINDOW_EVENT_MASK
);

}

public void processEvent (AWTEvent event) { // newButton
    long TS = 0;
    Button b,bb;
    double val;

    if (Debug==4)
        P(4,"processEvent.newButton",
            "(newButton)event="+event.paramString()+
            ", id="+event.getID());

    if (event.getID() == KeyEvent.KEY_PRESSED) {
        KeyEvent keyevent = (KeyEvent)event;
        int key = keyevent.getKeyCode();
        String keystring = keyevent.getKeyText(key);
        char ch = keyevent.getKeyChar();
        key.input(keyevent);

        // HERE
        //System.out.println("PRESSED keystring="+keystring+" keycode="+key+" keyevent("
        //+event.toString()+")");
        if (key == 16)
            is.shift_down = true;
        if (key == 17)
            is.control_down = true;
        if (key == 18)
            is.alt_down = true;
    }
    if (event.getID() == KeyEvent.KEY_RELEASED) {
        KeyEvent keyevent = (KeyEvent)event;
        int key = keyevent.getKeyCode();

        CREWS UP INPUT
        // HERE
        //System.out.println("RELEASED keystring="+keystring+" keycode="+key+" keyevent("
        //+event.toString()+")");
        if (key == 16)
            is.shift_down = false;
        if (key == 17)
            is.control_down = false;
        if (key == 18)
            is.alt_down = false;
    }
    if (event.getID() == MouseEvent.MOUSE_CLICKED) {
        TS = java.lang.System.currentTimeMillis();
        boolean is_meta_down = ((InputEvent)event).isMetaDown();

        if (is_meta_down)
            if (Debug==4)
                P(4,"newButton.processEvent","Meta down!!!");

            if (this == button_animate) {
                debug_animation_thread(1, "buttons", "this==button_animate");
                if (animation_thread == null) {
                    run_thread();
                } else {
                    stop_thread();
                }
            }
            if (this == button_presParent) {
                force.calcParentPressure = !force.calcParentPressure;
                graphics.setColor(Color.red);
                this.setLabel("Parent pres'="+force.calcParentPressure);
                if (force.calcParentPressure)
                    button_presParent.setBackground(color.obj.LightRed);
                else
                    button_presParent.setBackground(color.obj.LightGrey);
            }
            if (this == button_presPOV) {
                force.calcPOVPressure = !force.calcPOVPressure;
                graphics.setColor(Color.blue);
                this.setLabel("POV="+force.calcPOVPressure);
                if (force.calcPOVPressure)
```

```

    else
        button_presPOV.setBackground(color_obj.LightGrey);
    }

    if (this == button_presNeighbors)
    {
        force.calcNeighborPressure = !force.calcNeighborPressure;
        // graphics.setColor(Color.green);
        this.setLabel("Nbr="+force.calcNeighborPressure);
        if (force.calcNeighborPressure)
            button_presNeighbors.setBackground(color_obj.LightRed);
        else
            button_presNeighbors.setBackground(color_obj.LightGrey);
    }

    if (this == button_presNoise)
    {
        force.calcNoise = !force.calcNoise;
        this.setLabel("Noise="+force.calcNoise);
    }

    if (this == button_increase_mag, {
    }
    if (this == button_drawText) {
        mode_obj.toggle_draw_text();
    }
    if (this == button_absorb_POV) {
        dataseta.absorb_POV(false);
    }
    if (this == button_postprocessor) {
        dataseta.PostProcessor();
    }
    if (this == button_more_attraction) {
        val=dataseta.more_attraction(is_meta,down);
        this.setLabel("Mag Scale="+val);
    }
    if (this == button_more_repulsion) {
        val=dataseta.more_repulsion(is_meta,down);
        this.setLabel("ThetaMlt="+val);
    }
    if (this == button_lines) {
        mode_obj.toggle_lines_mode();
    }
    if (this == button_drawCN) {
        drawCN = !drawCN;
        this.setLabel("CN="+drawCN);
    }
    if (this == button_drawDN) {
        drawDN = !drawDN;
        this.setLabel("DN="+drawDN);
    }
}

// First time, create Dialog window. Subsequently create new node
if (this == button_background_color) { // SPREADING MODE
    color_obj.background_color_index++;
    if (color_obj.background_color_index >= color_obj.MAX_BACKGROUND_COLORS)
        color_obj.background_color_index=0;
    frame.setBackground(color_obj.background_color_array[color_obj.background_color_index]);
}
show_buttons();
}
if (this == button_position) { // POSITIONING MODE
    mode_obj.toggle_position_mode();
}
if (this == button_render) { // RENDERING MODE
    mode_obj.toggle_render_mode();
}
if (this == button_potentiation) { // POTENTIATION MODE
    mode_obj.toggle_potentiation();
    if (this == button_dump) {
        recursion_depth=0;
        dump.nodes(0, dataseta.Root); // arg is debug value
        // dump.node(0, Myself); // arg is debug value
    }
    if (this == button_Debug) {
        // See if left or right button is pressed
        if (is_meta,down)
            Debug--;
        else
            Debug++;
    }
}

// Set max/min for Debug
if (Debug > 5)
    Debug = 5;
if (Debug < -1)
    Debug = -1;
this.setLabel("Debug="+Debug);
}

if (this == button_Steop) {

```

```
quit();
```

```
    }  
    super.processEvent(event);  
} // class NewButton
```

```
/**  
**
```

```
*/  
class MyDialog extends Dialog {  
    TextField text;  
    public MyDialog(Frame f, String t) {  
        super(f, t);  
        setSize(200, 100);  
        text = new TextField("Hi, MyDialog creation ", 150);  
    }  
}
```

```
public void quit () {  
    try { java.lang.System.exit(0); }  
    catch (SecurityException e) { ; }  
} // end quit
```

```
/**  
*  
*  
*/
```

```
void add buttons () {  
    int x=80, y=20;
```

```
    int i = 0;  
    frame.add(button_animate = new JButton(" Start "));  
    button_animate.setBackground(color_obj.LightGrey);  
    button_animate.setForeground(Color.black);  
    button_animate.setLocation(90*i, 30);  
    button_animate.setSize(x, y);  
    button_animate.setVisible(true);
```

```
    i++;  
    frame.add(button_Stop = new JButton("EXIT"));  
    button_Stop.setBackground(Color.pink);  
    button_Stop.setForeground(Color.black);  
    button_Stop.setLocation(90*i, 30);  
    button_Stop.setSize(x, y);
```

```
button_Stop.setVisible(true);
```

```
    i++;  
    frame.add(button_Debug = new JButton("Debug="+Debug+" "));  
    button_Debug.setBackground(color_obj.LightGrey);  
    button_Debug.setForeground(Color.black);  
    button_Debug.setLocation(90*i, 30);  
    button_Debug.setSize(x, y);  
    button_Debug.setVisible(true);
```

```
    i++;  
    frame.add(button_render = new JButton("Node Rendering"));  
    button_render.setBackground(color_obj.LightGrey);  
    button_render.setForeground(Color.black);  
    button_render.setLocation(90*i, 30);  
    button_render.setSize(x, y);  
    button_render.setVisible(true);
```

```
    i++;  
    frame.add(button_background_color = new JButton("Background Colors"));  
    button_background_color.setBackground(color_obj.LightGrey);  
    button_background_color.setForeground(Color.black);  
    button_background_color.setLocation(90*i, 30);  
    button_background_color.setSize(x+30, y);  
    button_background_color.setVisible(true);
```

```
    i++;  
    frame.add(button_position = new JButton("Pos"));  
    button_position.setBackground(color_obj.LightGrey);  
    button_position.setForeground(Color.black);  
    button_position.setLocation(90*i, 30);  
    button_position.setSize(x, y);  
    button_position.setVisible(true);
```

```
    i++;  
    frame.add(button_potentiation = new JButton("Pot "+mode_obj.do_potentiation));  
    if (mode_obj.do_potentiation)  
        button_potentiation.setBackground(color_obj.LightRed);  
    else  
        button_potentiation.setBackground(color_obj.LightGrey);  
    button_potentiation.setForeground(Color.black);  
    button_potentiation.setLocation(90*i, 30);  
    button_potentiation.setSize(x, y);  
    button_potentiation.setVisible(false);  
    i++;  
    frame.add(button_Dump = new JButton("----- Dump Nodes-----"));  
    //button_Dump.setBackground(new Color(0xf0f0f0));
```

```
button_Dump.setBackground(color_obj.LightGrey);
button_Dump.setForeground(Color.black);
button_Dump.setLocation(90*i, 30);
button_Dump.setSize(x,y);
button_Dump.setVisible(false);
frame.add(check = new CheckBox("CheckBox. "));
check.setVisible(false);
```

```
i=0;
frame.add(button_drawText = new JButton("Inhibit Text"));
if (drawText)
    button_drawText.setBackground(color_obj.LightRed);
else
```

```
    button_drawText.setBackground(color_obj.LightGrey);
button_drawText.setForeground(Color.black);
button_drawText.setLocation(90*i, 60);
button_drawText.setSize(x,y);
button_drawText.setVisible(true);
```

```
i++;
frame.add(button_lines = new JButton("Lines=
"));
button_lines.setBackground(color_obj.LightGrey);
button_lines.setForeground(Color.black);
button_lines.setLocation(90*i, 60);
button_lines.setSize(x,y);
button_lines.setVisible(true);
```

```
i++;
frame.add(button_presParent = new JButton("Parent Pressure="+force.calcParentPressure)
```

```
);
if (force.calcParentPressure)
    button_presParent.setBackground(color_obj.LightRed);
else
```

```
    button_presParent.setBackground(color_obj.LightGrey);
button_presParent.setLocation(90*i, 60);
button_presParent.setForeground(Color.black);
button_presParent.setSize(x,y);
button_presParent.setVisible(true);
```

```
i++;
frame.add(button_presPOV = new JButton("POV="+force.calcPOVPressure));
if (force.calcPOVPressure)
    button_presPOV.setBackground(color_obj.LightRed);
else
    button_presPOV.setBackground(color_obj.LightGrey);
button_presPOV.setForeground(Color.black);
button_presPOV.setLocation(90*i, 60);
```

```
button_presPOV.setSize(x,y);
button_presPOV.setVisible(true);
i++;
frame.add(button_presNeighbors = new JButton("NeighborPressure="+force.calcNeighbor
Pressure));
```

```
if (force.calcNeighborPressure)
    button_presNeighbors.setBackground(color_obj.LightRed);
else
```

```
    button_presNeighbors.setBackground(color_obj.LightGrey);
button_presNeighbors.setForeground(Color.black);
button_presNeighbors.setLocation(90*i, 60);
button_presNeighbors.setSize(x,y);
button_presNeighbors.setVisible(true);
```

```
/******
```

```
i++;
frame.add(button_drawCN = new JButton("CN="+drawCN));
button_drawCN.setBackground(color_obj.LightGrey);
button_drawCN.setForeground(Color.black);
button_drawCN.setLocation(90*i - 65, 60);
button_drawCN.setSize(x/2,y);
button_drawCN.setVisible(false);
```

```
i++;
frame.add(button_drawDN = new JButton("DN="+drawDN));
button_drawDN.setBackground(color_obj.LightGrey);
button_drawDN.setForeground(Color.black);
button_drawDN.setLocation(90*(i-1), 60);
button_drawDN.setSize(x/2,y);
button_drawDN.setVisible(false);
```

```
frame.add(button_absorb_POV = new JButton("Absorb POV"));
button_absorb_POV.setBackground(color_obj.LightGrey);
button_absorb_POV.setForeground(Color.black);
button_absorb_POV.setLocation(90*i, 60);
button_absorb_POV.setSize(x,y);
button_absorb_POV.setVisible(false);
```

```
*****
```

```
i++;
frame.add(button_postprocessor = new JButton("PostProcessor"));
button_postprocessor.setBackground(color_obj.LightGrey);
button_postprocessor.setForeground(Color.black);
button_postprocessor.setLocation(90*i, 60);
button_postprocessor.setSize(x,y);
button_postprocessor.setVisible(false);
i++;
frame.add(button_reset = new JButton("Reset"));
button_reset.setBackground(color_obj.LightGrey);
```

05/23/00  
00:34:55

GUI.java

17

```

** button_reset.setBackground(Color.black);
** button_reset.setLocation(90*i, 60);
** button_reset.setSize(x,y);
** button_reset.setVisible(false);
** i++;
** frame.add(button_populate = new JButton("Populate. "));
** button_populate.setBackground(color_obj.LightGrey);
** button_populate.setForeground(Color.black);
** button_populate.setLocation(90*i, 60);
** button_populate.setSize(x,y);
** button_populate.setVisible(false);
** i++;
*****
** frame.add(button_more_attraction = new JButton("Screen Mag. "));
** button_more_attraction.setBackground(color_obj.LightGrey);
** button_more_attraction.setForeground(Color.black);
** button_more_attraction.setLocation(90*i, 30);
** button_more_attraction.setSize(x,y);
** button_more_attraction.setVisible(false);
** i++;
** frame.add(button_more_repulsion = new JButton("TheiaMilr. "));
** button_more_repulsion.setBackground(color_obj.LightGrey);
** button_more_repulsion.setForeground(Color.black);
** button_more_repulsion.setLocation(90*i, 30);
** button_more_repulsion.setSize(x,y);
** button_more_repulsion.setVisible(false);
** i++;
*****
** i++;
** frame.add(button_presNoise = new JButton("Noise="+force.calcNoise()));
** button_presNoise.setBackground(color_obj.LightGrey);
** button_presNoise.setForeground(Color.black);
** button_presNoise.setLocation(90*i, 90);
** button_presNoise.setSize(x,y);
** button_presNoise.setVisible(false);
*****/
}

mode_obj.set_mode_buttons();
}

```

```

** show_buttons
**
** public void show_buttons () {
**     System.err.println("show_buttons.");
**     button_animate.setBackground(color_obj.LightGrey);
**     button_animate.setBackground(Color.black);
**     button_Debug.setBackground(color_obj.LightGrey);
**     button_Debug.setBackground(Color.black);
**     button_render.setBackground(color_obj.LightGrey);
**     button_render.setBackground(Color.black);
**     button_render.setForeground(Color.black);
**     button_background_color.setBackground(color_obj.LightGrey);
**     button_background_color.setBackground(Color.black);
**     button_Stop.setBackground(color_obj.LightGrey);
**     button_Stop.setBackground(Color.black);
**     button_Stop.setForeground(Color.black);
**     button_drawText.setBackground(color_obj.LightGrey);
**     button_drawText.setBackground(Color.black);
**     button_lines.setBackground(color_obj.LightGrey);
**     button_lines.setBackground(Color.black);
**     button_presParent.setBackground(color_obj.LightGrey);
**     button_presParent.setBackground(Color.black);
**     button_presPOV.setBackground(color_obj.LightGrey);
**     button_presPOV.setBackground(Color.black);
**     button_presNeighbors.setBackground(color_obj.LightGrey);
**     button_presNeighbors.setBackground(Color.black);
** } // end show_buttons
**
** /*
** * map returns a Point in screen coordinates for rendering, considering
** * WindowOffsets and magscale
** */
** public void map (int x, int y, Point MapPoint) {
**     map((double)x, (double)y, MapPoint);
**     return;
** }
** public void map (double x, double y, Point MapPoint) {
**     double tx, ty;
**     double temp_x=0, temp_y=0;
**     tx = x - WindowXOffset;
**     tx *= magscale;
**     tx += WindowXcenter;
**     ty = y - WindowYOffset;
**     ty *= magscale;
**     tv = WindowYcenter - tv; // Y IS FLIPPED

```

```

// Now we have normal coordinates, consider TinyFlag
// get diff with embedded-universe node (e.g. Timeline), compress and add back to TL.x's
// if (TinyFlag && globalDoTiny)
/*****

if (TinyFlag) {
    if (TinyPoint != null) {
        temp_x = tx - TinyPoint.x;
        temp_y = ty - TinyPoint.y;
        temp_x *= TinyScale;
        temp_y *= TinyScale;
        tx = TinyPoint.x + temp_x;
        ty = TinyPoint.y + temp_y;
    }

    *****

    if (FlipAxes) {
        MapPoint.y = (int)tx;
        MapPoint.x = (int)y;
    }
    else {
        MapPoint.x = (int)tx;
        MapPoint.y = (int)ty;
    }

    return;
} // end map

/*
**
*
*
void clear_Tdist (Node parent){
Node tnode;
int i, size;

size = dataseta.node_vec.size();
for (i=0; i<size; i++) {
    tnode = (Node)dataseta.node_vec.elementAt(i);
    tnode.Tdist = -1;
}

return;
} // end clear_Tdist

} // end clear_Tdist

/*
**
*
*
void clear_dist (Node parent){
Node tnode;
int i, size;

size = dataseta.node_vec.size();
for (i=0; i<size; i++) {
    tnode = (Node)dataseta.node_vec.elementAt(i);
    tnode.dist = -1;
}

return;
} // end clear_dist

/*****

/*
* VRyes simple function to return whether to use VR mode or heirarchical mode
*/
static boolean VRyes (Node node) { // position this parent's children
boolean node_yes=false;

if (node == null) {
    ERROR(0, "VRyes", "Null node passed to me.");
    return(false);
}

if ((node.X != 0) || (node.Y != 0))
    node_yes = true;
else if (node == dataseta.POV)
    node_yes = true;

if (node.VRyes || (mode_obj.render_mode.equals("VR") && node_yes))
    return(true);
else
    return(false);
} // end VRyes

/**
** draw global str

```

```

**
*/
public void draw_global_str () {
    int i;
    Color current_color=null;

// DRAW GLOBAL STRING IF AVAILABLE

    if (global_str.size > 0)
    {
        current_color = graphics.getColor();
        graphics.setColor(color.obj.LightGrey);
        graphics.setColor(Color.black);
        // Limit printed results to 40 lines, else can get horrible slow
        global_str.size = (global_str.size>40) ? 40 : global_str.size;
        for (i=0; i<global_str.size; i++) {
            if (global_str[i] != null)
                graphics.drawString( global_str[i], 30,350+11*i );
            else
                ERROR(0, "draw_global_str", "NULL node global_str["+i+"]");
        }
        graphics.setColor(current_color);
    }

    // end draw_global_str

}

/**
** clear_status
**
*/
public void clear_status () {
    int i;
    StatusLine[0] = "-----";
    for (i=0; i<15; i++)
        StatusLine[i] = "";
    // end clear_status

}

/**
** draw_status
**
*/
public void draw_status () {
    int i;

    Color current_color=null;

// DRAW STATUS LINE STRING
    current_color = graphics.getColor();
    //graphics.setColor(color.obj.LightBlue);
    graphics.setColor(Color.black);

    graphics.drawString("nodes drawn: "+counter_of_positioned_nodes, 10, 68);
    graphics.drawString("frame:"+drawing_counter++, 20, 80);

    graphics.drawString("prior command:", 10,90);
    graphics.drawString("<"+lastCommand+">", 20, 102);
    graphics.drawString("current command:", 10, 114);
    graphics.drawString(key_input_string, 20, 126);

    graphics.drawString("Command History:", 10, 150);

    for (i=0; i<15; i++) {
        if (!StatusLine[i].equals(""))
            draw_string(StatusLine[i], 20, 160+12*i);
            //graphics.drawString( StatusLine[i], 20, 160+12*i);
    }
    graphics.setColor(current_color);

    return;
} // end draw_status

/**
*
*/
synchronized float position_start (Node parent) {
    if (parent == null)
        return(-1);

    if (DataSea.needdistUpdate) {
        datasea.set_dist_start(parent);
        datasea.set_ChildNum_start((Node)null, parent, datasea.theta.org);//false for theta.org fla
        datasea.needdistUpdate = false;
    }

    position_recursive((Node)null, parent); // POSITION

    return(0);
} // end position_start (a simple function calling recursive fns)

```

05/23/00  
00:34:55

# Confidential and Proprietary Property of Rocky Nevin

GUI.java

20

```
/**
 ** rescale
 **
 */
public void rescale (String cmd) {
    int i, size;
    Node node;
    Node child;
    double new_magscale, magdif;

    if (cmd.equalsIgnoreCase("clear")) {
        max_x = 20; // we want to always include the (0,0) cross-hairs
        min_x = -20;
        max_y = 20;
        min_y = -20;
        counter_of_positioned_nodes = 0;
    }
    else if (cmd.equalsIgnoreCase("run")) {
        double range_x = max_x - min_x;
        double range_y = max_y - min_y;
        double magx, magy;

        Window.Width=frame.getSize().width;
        Window.Height=frame.getSize().height;
        Window.Xcenter = Window.Width/2;
        Window.Ycenter = Window.Height/2;

        magx = Window.Width/range_x;
        magy = Window.Height/range_y;
        new_magscale = (magx<magy) ? magx : magy;

        // Slowly change offsets ...
        Window.XOffset += (int)(0.2*((min_x + max_x)/2.0 - Window.XOffset));
        Window.YOffset += (int)(0.2*((min_y + max_y)/2.0 - Window.YOffset));
        new_magscale *= 0.75; // Show all the edges

        // Now, slowly change magscale
        magdif = (new_magscale-magscale);
        if (magdif > 0.5) // don't allow it to zoom-in quickly
            magdif = 0.5;
        if (magdif < -1.60) // allow it to pull back quickly
            magdif = -1.60;
        magscale += magdif;
        /*****
        P(1,"rescale, run",min_x("+"min_x+""), max_x("+"max_x+""),magx="+"magx+"", magscale = "+"mags
        cale+", Window.XOffset = "+"Window.XOffset);
        P(1,"rescale, run",min_y("+"min_y+""), max_y("+"max_y+""),magy="+"magy+"", magscale = "+"mags
        cale+", Window.YOffset = "+"Window.YOffset);
        P(1,"rescale, run",Window.Xcenter="+"Window.Xcenter+", Y = "+"Window.Ycenter);
        *****/
    }
} // end rescale

/**
 ** random I have a feeling the seed is being reused ... 9/99 no, not true
 **
 */
static public double random () {
    //return(Math.random() + ((double)(java.lang.System.currentTimeMillis() % 10.0)/10.0);
    return(Math.random());
} // end random

/**
 ** flash
 **
 */
public void flash (Node node) {
    double x=random(), y=random();
    Point TempPoint=new Point();
    Color current_color=null;

    if (!datasea.gui_global.ok_to_draw)
        return;
    if (node==null)
        return;
    node.isSelected = true;
    show_node_once(node);
    node.isSelected = false;
    sleep(30);
    /*****
    x += node.x;
    y += node.y;
    maof(x.v, TempPoint);
    *****/
}
```

05/23/00  
00:34:55

## Confidential and Proprietary Property of Rocky Nevin

### GUI.java

21

```
graphics.setXORMode(Color.green);
//current_color = graphics.getColor();
//graphics.setColor(Color.red);
graphics.drawLine(TempPoint.x, TempPoint.y, TempPoint.x+2, TempPoint.y); // draw a fake chr
osome
//graphics.setColor(current_color);
graphics.setPaintMode();
*****
) // end flash

/**
 ** OKtoRenderRecursive
 **
 */
public boolean OKtoRenderRecursive (Node parent, Node child) {
boolean continue_flag=false;

if (GUI.mode_obj.position_mode.equals("Lvs") ||
    GUI.mode_obj.position_mode.equals("Grd"))
    return(true);

if (parent == datasea.POV)
    return(true);
if (child == datasea.POV)
    return(true);
if (child.dist <= 4)
    return(true);

// Make sure we should render child, based on dist, then recurse
if (child == null) {
    WARNING(0, "GUI.OKtoRenderRecursive", "NULL CHILD, RETURNING, parent="+parent.
    Name+"(dist="+
    +parent.dist+"), child="+child.Name+"(dist="+child.dist+"");
    return(false);
}

// Don't draw more than once per 'paint()'
if (child.drawnTS == current_TS) {
    return(false);
}
else

    child.drawnTS = current_TS;

if (child.dist == -1) { // We'll never see this or its children if its not linked to POV
    return(false);
}

if (child.isEvent && drawEvent)
    continue_flag |= true;
if (child.isCN && drawCN)
    continue_flag |= true;
if (child.isAN && drawAN)
    continue_flag |= true;
if (child.isON && drawON)
    continue_flag |= true;
if (child.isDN && drawDN)
    continue_flag |= true;
if (child.isFile && drawFile)
    continue_flag |= true;
if (child.isURL && drawURL)
    continue_flag |= true;

if (!continue_flag)
    return(false);
if (child.mag < relations.threshold && !child.isSelected) {
    return(false);
}
if (parent != null && child.isPOV) { // Don't draw towards a POV
    return(false);
}
if ((parent != null) && (child == datasea.Root)) { // don't draw Root as anyone's child
    return(false);
}
return(true);
} // end OKtoRenderRecursive

/*
 *
 */
float render_start (Node parent){
if (parent == null)
    return(-1);
clear();
```

05/23/00  
00:34:55

## Confidential and Proprietary Property of Rocky Nevin GUI.java

22

```
graphics.setColor(color_obj.LightGrey);
if (global_str.size > 0)
    draw_global_str();
draw_status();
render_recursive(null, parent); // DRAW
return(0);
} // end render_start (a simple function calling recursive fns)

/*
 * render_recursive checks VBytes(child) and calls rendering fn's, then recurses
 */
public void render_recursive (Node parent, Node child) { //
    int i, size;
    Node grand_child, CNode;
    Point ChildPoint=new Point();
    double prior_max_mag=0;
    boolean PoLOK = false;
    boolean OK=true;

    OK = OKtoRenderRecursive(parent, child);

    if (quick && !OK) {
        return;
    }

    // temp override
    color_obj.setColor_from_TS(graphics, child);
    render_node(parent, child, false);
    /*****
    if (xor)
        graphics.setXORMode(Color.white);
    else
        if (child.isSelected == true)
            graphics.setColor(Color.red);
        else
            color_obj.setColor_from_mag(graphics, child);
    *****/

    // MAKE RECURSIVE CALLS ON CHILDREN
    size = child.Links.size();
    for (i=0; i<size; i++) { // check dist of all children, position if appropriate

        // This is dumb, but I have trouble controlling threads and synchronizing blocks
        /*****
        if (child == null) // bad synchronization between rendering and linking threads
            break;
        if (i >= child.Links.size()) // bad synchronization between rendering and linking threads
            break;
        /*****
        grand_child = child.getNodeAtLink(i);

        // 4/29/2000
        if (grand_child == null)
            continue;

        if (OK) { // don't draw if not OK, but do recurse nonetheless

            // 4/29/2000
            if (quick) {
                if (grand_child.mag < Node.MED_MAG && child.mag < Node.MED_MAG) {
                    //System.out.print("R["+grand_child.Name+"] ");
                    continue;
                }
            }

            // See if we should render link--modulating CN's
            if (drawCN) {
                CNode = child.getCNodeAtLink(i);
                if (CNode != null) {
                    //P(2,"render_recursive",
                    //    "["+i+"] CNode of "+child.Name+" to "+grand_child.Name+" is "+C
                    //    Node.Name);
                    render_node(child, CNode, false);
                }
            }
            } // if OK
        }
        if (grand_child.hasLargerDistThan(child)) { // recurse regardless of OK
            render_recursive(child, grand_child);
        }
        } // for i
    return;
    } // end render_recursive

/*
 *
 */
void position_recursive (Node parent, Node child) { // position this parent's children
    int i, size;
    boolean PoLOK = false;
```

05/23/00  
00:34:55

# Confidential and Proprietary Property of Rocky Nevin

GUI.java

23

Node grand\_child, CNode=null;

```
if (child == null) {
    WARNING(0,"position_recursive", "child is null");
    return;
}

if (child.dist < 1) {
    //WARNING(0,"position_recursive", "child dist <1"+child.Name);
    return;
}

//size = child.Links.size();
size = child.ChildCount;
for (i=0; i<size; i++) { // position and then recurse on all children
    // ordered list, set in set_ChildNum_recursive()
    grand_child = child.getNodeAtLink(child.vec[i]);
    if (grand_child == null) {
        return;
    }
    if (grand_child.isAN && !drawAN) {
        return;
    }
    if (grand_child.isEvent && !drawEvent) {
        return;
    }
}

// 4/19/2000
if (quick) {
    if (grand_child.mag < Node.MED_MAG-1 && child.mag < Node.MED_MAG-1) {
        //System.out.println("P"+grand_child.Name+"");
        return;
    }
}

if (grand_child.dist > child.dist) {
    CNode = child.getCNodeAtLink(i);
    if (CNode != null)
        position_node(child, CNode);
    position_node(child, grand_child);
    position_recursive(child, grand_child);
}

return;
} // end position_recursive
```

```
/*
** METHOD position_node(parent, child) sets the child's x and y vars into
** screen coordinates based on it's parent's position
**
*/
float position_node (Node parent, Node child) {
    double pxSum=0, pySum=0;
    Node tn=null;

    if (parent == null)
        return(0);
    if (child == null)
        return(0);

    if (child.Debug && Debug==1) {
        System.out.println("position_node(): parent=<"+parent.Name+"> -> child=<"+child.Name>);
        if (parent.Debug && Debug==1) {
            System.out.println("position_node(): parent=<"+parent.Name+"> -> child=<"+child.Name>);
        }
    }

    // RUNNING SET OF MAX AND MIN VALUES OF COORDINATES HERE

    counter_of_positioned_nodes++;
    if (min_x > child.x)
        min_x = child.x;
    if (min_y > child.y)
        min_y = child.y;
    if (max_x < child.x)
        max_x = child.x;
    if (max_y < child.y)
        max_y = child.y;

    if (child.isLayer) {
        child.x = child.X;
        child.y = child.Y;
        return(0);
    }

    if (VRyes(child)) { // draw in absolute position, not relative to caller
        child.x = (0.9*child.X+0.1*child.x); // move 90% each update to ideal spot
    }
```

```

        child.y = (0.9*child.Y+0.1*child.y); // move 90% each update to ideal spot
    }
    else {
        force.calc.pressures(parent,child);
        if (globalMaxPressure > 10) {
            pxSum = 10*force.pxSum/globalMaxPressure;
            pySum = 10*force.pySum/globalMaxPressure;
        }
        else
        {
            pxSum = force.pxSum;
            pySum = force.pySum;
        }
        child.px = pressure_mag*pxSum*Math.abs(pxSum);
        child.py = pressure_mag*pySum*Math.abs(pySum);
        child.x += child.px;
        child.y += child.py;
        if (Debug > 0) {
            child.pxParent = force.pxParent;
            child.pxPOV = force.pxPOV;
            child.pxNeighbors = force.pxNeighbors;
            child.pyParent = force.pyParent;
            child.pyPOV = force.pyPOV;
            child.pyNeighbors = force.pyNeighbors;
        }
    }
    if (drawEvent) {
        if (child.isEvent && parent.isDN) { // force DN's near their event
            parent.x = child.y + 50 + 10/magscale;
            parent.y = child.y+2 + 2.0*(parent.ChildNum-3);
            parent.ThetaOffset = 0;
        }
        else
        if (parent.isEvent && child.isDN) { // force DN's near their event
            child.x = parent.x + 50 + 10/magscale;
            child.y = parent.y+2 + 2.0*(child.ChildNum-3);
            child.ThetaOffset = 0;
        }
        else
        if (child.isDN && (null != (tn=child.hassiblingOfType("Event")))) { // else force DN's near any related event
            child.x = tn.x + 70 + 10/magscale;
            child.y = tn.y+2 + 1.5*(child.ChildNum-3);
            child.ThetaOffset = 0;
        }
    }
}

return(0);
} // end position_node

/**
 * method nsr_render_chromosome
 */
boolean nsr_render_chromosome (Node node) {
    int x1, y1, x2, y2, t;
    x1 = (int)(node.x - magscale*10) + WindowXcenter;
    y1 = (int)(node.y - magscale*30) + WindowYcenter;
    x2 = (int)(node.x + magscale*10) + WindowXcenter;
    y2 = (int)(node.y + magscale*30) + WindowYcenter;
    graphics.drawLine(x1, y1, x2, y2); // draw a fake chromosome
    graphics.drawLine(x1, y1, x1, y1-(int)(magscale*5));
    t=x1;
    x1=x2;
    x2=t;
    graphics.drawLine(x1, y1, x2, y2); // draw a fake chromosome
    graphics.drawLine(x1, y1, x1, y1-(int)(magscale*5));
    return(true);
} // end nsr_render_chromosome

/**
 * trim return a string of 'length' precision of argument double 'x'
 */
public String millis (long x) {
    String s;
    int len=0;
    s = java.lang.Long.toString(x);
    len = s.length()-3;
    if (len < 0)
        len = 0;
    return((String)(s.substring(0,len)));
} // end trim

/**
 * prec return a string of 'length' precision of argument double 'x'
 */
public String prec (double x, int length) {
    String s;
    int max_length, printed_length;;
    s = java.lang.Double.toString(x);

```

```

        max_length = s.length();
        printed_length = (max_length < length ? max_length : length;
        return((String)s.substring(0,printed_length));
    } // end prec

/**
 ** state
 **
 */
    public void state () {
        //if (s.equalsIgnoreCase("magscale"))
        P(0,"state", "magscale = "+magscale);
        //if (s.equalsIgnoreCase("window"))
        P(0,"state", "WindowXcenter= "+WindowXcenter+" WindowYcenter= "+WindowYcenter+" WindowXOffset= "+WindowXOffset+" WindowYOffset= "+WindowYOffset);
    } // end state

/*****
 **
 ** draw_string
 **
 */
    public void draw_string (String s, Point p) {
        draw_string(s, p.x, p.y);
    } // end one version of draw_string

/*****
 public void draw_string (String s, int x, int y) {
     int i=0, len=s.length(), start=0, end=0;
     String ss;

     // frame.setFont(titleFont);
     end = s.length();
     start = 0;

     if (end <= len)
         graphics.drawString(s, x, y);
     else
         while (start < end) {
             line_len = end - start;
             if (line_len > len)
                 line_len = len;
             if ((s.substring(start+line_len-1, start+line_len).equals(" ")) ||

```

---

```

        (start+line_len == end))
            ss = s.substring(start, start+line_len);
        else
            ss = s.substring(start, start+line_len)+"-";
        graphics.drawString(ss, x, y+(10*i++));
        start += line_len;
    }
    // frame.setFont(titleFont);

    } // end draw_string
/*****
 **
 ** method show_node_once
 **
 */
    void show_node_once (Node node) {
        boolean selected;
        if (node == null) {
            WARNING(0, "show_node_once", "NULL node");
            return;
        }

        // We must switch the graphics context to draw on the live image immediately
        // rather than in double-buffer mode, so use image.counter and handle it.
        graphics = frame.getGraphics();

        render_node(null, node, true); // draw immediately, 'true' refers to XOR mode
        if (image.counter == 1) { // now undo the damage to the graphics context 'graphics'
            graphics = image1.getGraphics();
        } else {
            graphics = image2.getGraphics();
        }

        return;
    } // end show_node_once

/*****
 * method render_node
 */
    float render_node (Node parent, Node child, boolean xor) {
        Node node = null;
        int i, size;

```

// CHILD ChildPoint is the upper left corner in screen coordinates

```

if (child.isForm) {
    graphics.fillRect(ChildPoint.x, ChildPoint.y-height, width, height, true);
}
else if (child.isAN)
    graphics.drawOval(ChildPoint.x, ChildPoint.y-height, width, height);
else if (child.isEven) {
    graphics.fillRect(ChildPoint.x, ChildPoint.y-height, width, height, true);
}
else if (child.Type.equals("TL")) {
    graphics.fillRect(ChildPoint.x, ChildPoint.y-height, width, height, true);
}
else if (drawBoxes && child.isDN)
    graphics.drawRect(ChildPoint.x, ChildPoint.y-height, width, height);
else if (drawBoxes && child.isURL) {
    current_color = graphics.getColor();
    if (child.Data[0]==null) {
        graphics.setColor(color_obj.Grey);
    } else {
        graphics.setColor(color_obj.VeryLightGrey);
    }
}
graphics.fillRect(ChildPoint.x, ChildPoint.y-height,
    (int)(0.7*magscale*width), (int)(magscale*height), true);
graphics.setColor(current_color);
}
}
if (child.isCN) //superimpose oval if acting as a CNode

```

```

        current_color = graphics.getColor();
        graphics.setColor(color_obj.VeryLightRed);
        graphics.drawOval(ChildPoint.x, ChildPoint.y-height,
            (int)(2*width), (int)(height));
        graphics.setColor(current_color);
    }

else
    graphics.drawLine(ChildPoint.x, ChildPoint.y-height, ChildPoint.x+4, ChildPoint.y-height);
    if (Details) {
        if (
            ((child.x - spaceX) > -5) &&
            ((child.x - spaceX) < 5) &&
            ((child.y - spaceY) > -5) &&
            ((child.y - spaceY) < 5) &&
            (child.mag >= box_threshold)
        )
            ZoomOn = true;
        else
            ZoomOn = false;
    }
    // DRAW NAME
    if (((Debug == 1)&&(child.mag>text_threshold)) || (ZoomOn)) {
        // Mag, distance
        graphics.drawString("["m(" + prec(child.mag,3)
            + ") d(" + child.dist + ")"]"
            + child.Name
            + ", TS=" + $TS + "]",
            ChildPoint.x, ChildPoint.y+10);
    }
    /*****
    graphics.drawString(child.Name+", Description="+child.Desc+"", ChildPoint.x+width
    +1, ChildPoint.y);
    *****/
else
    if (drawText) {
        if (ldrawDN && child.isDN)
            ;
        else {
            if (child.mag > text_threshold) {
                if (child.isURL) {
                    //graphics.setColor(Color.blue);
                    i = 1 + child.Name.lastIndexOf("/"); // add one to work with substrin
                    if (i > 0) {
                        //base = child.Name.substring(0,i);
                        //name = child.Name.substring(i);
                        draw_string(child.Name.substring(i), ChildPoint.x+width+
                            1, ChildPoint.y);
                    }
                }
                else
                    draw_string(child.Name, ChildPoint.x+width+1, ChildPoint.y);
            }
        }
    }
    // PARENT
    if (parent != null) {
        if (child == datasea.Root)
            return(0); // Bail if we are trying to draw the root from another node
        map(parent.x, parent.y, ParentPoint);
    }
    // DRAW RELATIONS AS LINES
    draw_relations(child, parent, xor);
    // In this block, draw relations in yellow from distal ANs
    {
        size = child.Links.size();
        for (i=0; i<size; i++) { // check dist of all children, position if appropriate
            inode = child.getNodeAtLink(i);
            if (inode.isAN && inode != parent)
                draw_relations(child, inode, xor);
        }
    }
    // END DRAW RELATIONS
    if ((parent == datasea.Root) || (parent == datasea.POV)) // Special case, else we won't see it
        graphics.drawString(parent.Name, ParentPoint.x, ParentPoint.y);
    }
    // RADIAL LINES FOR LASTNODE
    if (child == lastNode)
    {
        if (FlipAxes) {
            child_center_y = (int)(ChildPoint.x + width/2);
            child_center_x = (int)(ChildPoint.y + height/2);
        }
    }
}

```

05/23/00  
00:34:55

## Confidential and Proprietary Property of Rocky Nevin

GUI.java

29

```
    }
    else {
        child_center.x = (int)(ChildPoint.x + width/2);
        child_center.y = (int)(ChildPoint.y - height/2);
    }

// PRESSURE LINES
if (Debug > 0) {
    graphics.setColor(Color.blue);
    graphics.drawLine(child_center.x, child_center.y,
        (int)(child_center.x+10*child.pxPOV), (int)(child_center.y-10*child.pyPOV));
    graphics.setColor(Color.red);
    graphics.drawLine(child_center.x, child_center.y,
        (int)(child_center.x+10*child.pxParent), (int)(child_center.y-10*child.pyParen
    ));
    graphics.setColor(Color.green);
    graphics.drawLine(child_center.x, child_center.y,
        (int)(child_center.x+10*child.pxNeighbors), (int)(child_center.y-10*child.pyN
    eighbors));
}
else { // OR MARK WITH AN X
    graphics.drawLine(child_center.x+10, child_center.y+10, child_center.x+30, child_center.y+30);
    graphics.drawLine(child_center.x-10, child_center.y+10, child_center.x-30, child_center.y+30);
    graphics.drawLine(child_center.x-10, child_center.y-10, child_center.x-30, child_center.y-30);
    graphics.drawLine(child_center.x+10, child_center.y-10, child_center.x+30, child_center.y-30);
}
} // end if child == lastNode
if (xor)
    graphics.setPaintMode();
return(0);
} // end render_node

/**
 ** draw_relations
 **
 */
public void draw_relations (Node child, Node parent, boolean xor) {
    int i, size;
    Node mode;
    Link link=null;
    Color current_color=null;
    Point ParentPoint=new Point(), ChildPoint=new Point(), TempPoint=new Point();
    if (child==null || parent==null)

        return;
    map(child.x, child.y, ChildPoint);
    map(parent.x, parent.y, ParentPoint);

// DRAW RELATIONS AS LINES
//if ((parent.isCN || child.isCN) && !drawCN)
//
//    if ((parent.isAN || child.isAN) && !drawAN)
//
//        else if ((parent.isDN || child.isDN) && !drawDN)
//
//            else if (mode.obj.lines.mode.equals("none"))
//
//                else if (child == datasea.Root)
//
//                    else if ((child.mag >= relations.threshold) && (parent.mag >= relations.threshold)) {
//                        if (xor)
//                            graphics.setXORMode(Color.green);
//                        else
//                            color.obj.set_color_for_relations(graphics, parent, child);
//                        if (mode.obj.lines.mode.equals("local")) {
//                            if (lastNode != null) {
//                                if ((lastNode == parent) || (lastNode == child)) {
//                                    // graphics.drawLine(ChildPoint.x, ChildPoint.y,
//                                    // ParentPoint.x, ParentPoint.y);
//                                    size = child.links.size();
//                                    for (i=0; i<size; i++) { // check dist of all children, position if appropriate
//                                        mode = child.genNodeAtLink(i);
//                                        map(mode.x, mode.y, TempPoint);
//                                        graphics.drawLine(TempPoint.x, TempPoint.y,
//                                            ChildPoint.x, ChildPoint.y); // child to its siblings
//                                    }
//                                }
//                            }
//                        }
//                    }
//                else if (mode.obj.lines.mode.equals("all")) {
//                    if (child.isSelected == true) // special case for traceback
//                        graphics.setColor(Color.red);
//                    boolean is_aliased_time = false;
//                    if (child==datasea.pod.yesterday node || child==datasea.pod.today node || child==datasea
```

05/23/00  
00:34:55

# Confidential and Proprietary Property of Rocky Nevin

GUI.java

29

```

a.pop tomorrow_node
    ll child==datasea.pop.last_week_node ll child==datasea.pop.this_week_node ll ch
id==datasea.pop.next_week_node
    is_aliased_time = true;

    if (child.isEvent && parent.isEvent && !is_aliased_time)
        graphics.drawLine(ChildPoint.x, ChildPoint.y,
            ParentPoint.x, ChildPoint.y); // Event to Timeline
    else
        if (parent != datasea.POV)
            graphics.drawLine(ChildPoint.x, ChildPoint.y,
                ParentPoint.x, ParentPoint.y); // child to parent
        }

    if (drawLinkNames
        && ((parent.mag > Node.BIG_MAG && child.mag > Node.BIG_MAG)
            ll (parent==gui.lastNode ll child==gui.lastNode))) {
        int x1=0, y1=0, x2=0, y2=0;

        link = parent.getLinkTo(child);
        if (link != null) {
            if (link.Name != "")
                graphics.drawString(" "+link.Name+""),
                    (int)((ParentPoint.x+ChildPoint.x)/2),
                    (int)((ParentPoint.y+ChildPoint.y)/2));
            if (parent.getPol(child) == '+') {
                x1 = (int)ParentPoint.x;
                y1 = (int)ParentPoint.y;
                x2 = (int)(ParentPoint.x+(ChildPoint.x-ParentPoint.x)*0.7);
                y2 = (int)(ParentPoint.y+(ChildPoint.y-ParentPoint.y)*0.7);
                graphics.drawOval( x2, y2, 3, 3);
                //graphics.drawLine( x1, y1, x2, y2);
                // child to parent
            }
        }
    else
        if (parent.getPol(child) == '-') {
            x1 = (int)ParentPoint.x;
            y1 = (int)ParentPoint.y;
            x2 = (int)(ParentPoint.x+(ChildPoint.x-ParentPoint.x)*0.3);
            y2 = (int)(ParentPoint.y+(ChildPoint.y-ParentPoint.y)*0.3);
            graphics.drawOval( x2, y2, 3, 3);
            //graphics.drawLine( x1, y1, x2, y2);
        }
    else {
        x1 = (int)ParentPoint.x;
    }

    y1 = (int)ParentPoint.y;
    x2 = (int)(ParentPoint.x+(ChildPoint.x-ParentPoint.x)*0.7);
    y2 = (int)(ParentPoint.y+(ChildPoint.y-ParentPoint.y)*0.7);
    graphics.drawOval( x2, y2, 5, 3);
    //graphics.drawLine( x1, y1, x2, y2);
}

}

// Special case drawing of relations, if isSelected
if (child.isSelected) {
    current_color = graphics.getColor();
    size = child.links.size();
    for (i=0; i<size; i++) { // check dist of all children, position if appropriate
        inode = child.getNodeAtLink(i);
        map(inode.x, inode.y, TempPoint);

        // set the colors
        if (inode.dist == child.dist)
            graphics.setColor(Color.green);
        else
            if (inode.dist >= child.dist+0.1)
                graphics.setColor(Color.blue);
            else
                if (inode.dist == child.dist-1)
                    graphics.setColor(Color.red);
                else
                    graphics.setColor(Color.yellow); // not within dist of 1
                    graphics.drawLine(TempPoint.x, TempPoint.y,
                        ChildPoint.x, ChildPoint.y); // child to its siblings
            }
        graphics.setColor(current_color);
    }

    // mag thresholds
} // end draw_relations

/**
 * method createMyself Creates a node, links it to 'this'
 */

```

```
public void createMyself () {
    if (GlobalNodeNode == 0) {
        GlobalNodeNode ++;
        Myself = new Node("Node", "Java Object", "The Essential Node");
        P(0, "GUI.createMyself", "Initializing the Myself Object");
        Myself.describe("To Console");
    }
} // end createMyself

/**
 ** demo1
 */
public void demo1 () {
    int i;
    Node a, b;

    P(0, "demo1", "Begin");
    input.string_input("show thes");
    input.string_input("3");
    input.string_input("poimag egg art");
    input.string_input("poimag egg art");
    input.string_input("poimag egg art");

    P(0, "demo1", "Done.");
} // end demo1

/**
 ** demo2
 */
public void demo2 () {
    int i;

    } // end demo2

/**
 ** demo2
 */
public void demo3 () {
    } // end demo3

/**
 ** method run_thread
 */
public void run_thread () {

    Thread active_thread;

    active_thread = Thread.currentThread();

    if (animation_thread != null) {
        WARNING(0, "run_thread", "Starting thread, old one exists. "+
            "animation_thread = "+animation_thread+"");
        animation_thread.start();
        button_animate.setLabel("Running");
        return;
    }

    debug_animation_thread(1, "run_thread", "animation thread is null, creating new Thread(this)");
    animation_thread = new Thread(this);
    animation_thread.start();
    // should invoke run() below

    button_animate.setLabel("Running");

    debug_animation_thread(1, "run_thread", "after new Thread(this)");

    P(0, "run_thread", "Thread started: "+animation_thread);
    } // end run_thread

/**
 ** debug_animation_thread
 **
 */
    static public void debug_animation_thread (int debug_level, String fn_name, String fn_desc) {
    c) {

        if (animation_thread == null)
            P(debug_level, "debug_animation_thread["+fn_name+"]: "+fn_desc+"<null, activeCount="+
                +Thread.activeCount()+ "+Thread.currentThread()>");
        else {
            P(debug_level, "debug_animation_thread["+fn_name+"]: "+fn_desc+"<isAlive="+animati
                on_thread.isAlive()+", activeCount="+Thread.activeCount()+ "+Thread.currentThread()>");
        }

    } // end debug_animation_thread

} // end debug_animation_thread

/**
```

05/23/00  
00:34:55

## Confidential and Proprietary Property of Rocky Nevin

GUI.java

31

```
** method stop_thread
**
** This should only be called from a point guaranteed not to be
** inside a routine recursively following the node network, e.g.
** from update() which checks StopThreadRequest to see if we
** should be called.
*/
public void stop_thread () {
    Thread active_thread;

    P(0, "stop_thread", "stopping animation_thread");

    if (animation_thread != null) {
        if (animation_thread.isAlive()) {
            P(0, "stop_thread", "isAlive(), calling animation_thread.stop(), then nulling it");
            animation_thread.stop();
            animation_thread = null;
        }
        else
            debug_animation_thread(1, "stop_thread", "not alive, not stopping it...");
    }

    /******
    // try ( animation_thread.join(1000); )
    // catch (InterruptedException i) {
    //     ERROR(0, "stop_thread", "ERROR FROM 'join(1000)', InterruptedE
    //         xception "+i);
    // }
    // animation_thread.stop();
    /******
    }
    else
        P(0, "stop_thread", "animation_thread is null");
    debug_animation_thread(1, "stop_thread", "apparently animation_thread is null");

    button_animate.setSelected("Stopped");
    //datasea.needitistUpdate = true;
    } // end stop_thread

/**
** start
**
*/
public void start () {
    // Applet is started, don't do anything special
    P(0, "start", "ran.");

    }

    public void stop () {
        // Applet is stopped, stop thread also
        if (animation_thread != null) {
            if (animation_thread.isAlive()) {
                P(0, "stop", "isAlive so running stop_thread()");
                stop_thread();
            }
            else {
                P(0, "stop", "ran, 'animation_thread' = null");
            }
        } // end start

    }

    /**
    ** sleep
    **
    */
    public static void sleep (int millis) {
        try { Thread.sleep(millis); }
        catch (InterruptedException e) { ; }
    } // end sleep

    /**
    ** run
    **
    */
    public void run () {
        // This is for the thread animation_thread
        int i;
        date = new java.util.Date();
        timestamp = new Timestamp(date.getTime());

        System.err.println("---- run(), BEGUN ----");
        while (true) {
            sleep(10);
            update(true);
            if (StopThreadRequest) {
                Animating = false;
                System.err.println("---- run(), StopThreadRequest ----");
                break; // thread should be done by leaving this function
            }
        }

    }

    /**
    ** i convert to integer
    **
    */
```

05/23/00  
00:34:55

# Confidential and Proprietary Property of Rocky Nevin

GUI.java

82

```
*/
    public static int i (float x) {
        return ( (int)x );
    }

    public static int i (double x) {
        return ( (int)x );
    }

/**
 ** status
 **
 */
    static public void status (String string) {
        int i;

        for (i=14; i>0; i--)
            StatusLine[i] = StatusLine[i-1];
        StatusLine[0] = string;

    } // end status

/**
 ** WARNING    Print WARNING messages
 **
 */
    public static void WARNING (int debug_thresh, String fn, String string) {
        if (debug_thresh <= Debug) {
            if (list != null)
                list.add("==> WARNING in "+fn+": " + string + " ", 0);
            System.err.println("==> WARNING in "+fn+": " + string);
            // debug_animation_thread(1, fn, string);
            status("WARNING: " + string);
        }
    } // end WARNING

/**
 ** ERROR    Print ERROR messages
 **
 */
    public static void ERROR (int debug_thresh, String fn, String string) {
        if (debug_thresh <= Debug) {
            beep();
            if (list != null)

list.add("==== ERROR in function "+fn+": " + string + " ", 0);
            System.err.println("==== ERROR in function "+fn+": " + string);
            dump_stack(fn);
            global_str_size = 1;
            global_str[0] = "ERROR in fn "+fn+": " + string;
            sleep(3000) // Erase this after a bit ...
            global_str_size = 1;
            global_str[0] = "(prior error was in fn "+fn+": " + string + ")";
            status("ERROR: " + string);
        }
    }

/**
 ** P    Print debug messages if given value == current Debug value
 **
 */
    public static void P (int debug_thresh, String fn, String string) {
        if (debug_thresh == Debug) {
            if (list != null) {
                list.add(" "+fn+": " + string, 0);
                System.err.println(" "+fn+": " + string);
            }
            //status(string);
        }
    }

/**
 ** dump_stack
 **
 */
    static public void dump_stack (String fn_name) {
        System.err.println("Dumping current Thread's stack trace deliberately, by "+fn_name);
        System.err.println("dump_stack(): " + "=====");
        (Thread.currentThread()).dumpStack();
        System.err.println("dump_stack(): " + "=====");
    } // end dump_stack

/**
 ** method print tree
 **
```

```

    } // end add_to_global_str

    /**
     ** print_tree_data.start
     **
     */

    public void print_tree_data.start (Node node) {
        int i, j, size, lines_count=1;
        Node tn;

        clear_global_str();

        size = node.links.size();
        add_to_global_str("-----");

        add_to_global_str(node.Name+"("+node.Type+", "+node.Desc+"); "+node.Data);
        for (i=0; i<size; i++) {
            tn = node.getNodeAtLink(i);
            //if (tn.dist == node.dist+1)
                if (tn.dist > node.dist)
                    lines_count += print_tree_data_recursive(tn);
        }

        add_to_global_str("-----");
        add_to_global_str("Lines returned = "
            +lines_count+"", global_str.size="+global_str.size);
    } // end print_tree_data.start

    /**
     ** print_tree_data_recursive
     **
     */

    public int print_tree_data_recursive (Node node) {
        int i, j, size, spaces_count, line_count=1;
        Node tn;
        String spaces;
        String s=null;

        spaces_count = 60-(int)(node.mag*10.0);
        spaces = "";
        for (i=0; i<spaces_count; i++) // concatenate spaces_count " "
            spaces = spaces + " ";

        if (node.mag > 1)

```

```

g.3)+"");
    size = node.Links.size();
    for (i=0; i<size; i++) {
        tn = node.getNodeAtLink(i);
        //if (tn.dist == node.dist+1)
            if (tn.dist > node.dist)
                line_count += print_tree_data_recursive(tn);
    }
    return(line_count);
} // end print_tree_data_recursive

/**
 ** dump_data
 **
 */
    public void dump_data (int debug_value, Node node) {
        int i, j, size;
        P(debug_value, "dump_data", node.Name+" Desc="+node.Desc+", Data="+node.Data);
    }

/**
 ** dump_node
 **
 */
    public void dump_node (int debug_value, Node node) {
        int i, j, size;
        String str;
        Node tnode;
        Link link=null;

        if (Debug < debug_value)
            return;

        if (node == null) {
            ERROR(0, "dump_node", "node is null");
            return;
        }
        size = node.Links.size();
        for (j=0; j<recursion_depth; j++) {

            System.err.println("-----");
            str = "dmp:" + node.Name
                + " dist=" + node.dist
                + ",Tdist=" + node.Tdist
                + ",mag=" + prec(node.mag.3)
                + " " + node.Type + " "
                + " isEvent=" + node.isEvent + " "
                + " isDN=" + node.isDN + " "
                + " isAN=" + node.isAN + " "
                + " isFile=" + node.isFile + " "
                + " isDirectory=" + node.isDirectory + " "
                + " isCN=" + node.isCN + " "
                + " isURL=" + node.isURL + " "
                + " ChildNum=" + node.ChildNum
                + " BigChildCount=" + node.BigChildCount
                + " (x,y)=( " + (node.x) + " " + (node.y) + ")"
                + " (size_X,size_Y)=( " + (node.size_X) + " " + (node.size_Y) + ")"
                + " (X,Y)=( " + (node.X) + " " + (node.Y) + ")"
                + " (size_X,size_Y)=( " + (node.size_X) + " " + (node.size_Y) + " )";
            list.add(str, 0);
            System.err.println(str);
        }
        if (show_links)
        {
            for (i=0; i<size; i++)
            {
                tnode = node.getNodeAtLink(i);

                if (tnode == node)
                    System.out.println("dump: node=child, <"+node.Name+">");
                str = " "
                    + tnode.Name + " "
                    + " dist=" + tnode.dist
                    + " ,Tdist=" + tnode.Tdist
                    + " ,mag=" + prec(tnode.mag.3)
                    + " " + tnode.Type + " "
                    + " CS(" + prec(node.get_CS(tnode.3)) + ")"
                    + " ,ChildNum=" + tnode.ChildNum
                    + " ,BigChildCount=" + tnode.BigChildCount;
                list.add(str, 0);
                System.err.println(str);
            }
        }
        if (node.isCN)
        {
            size = node.ContextLinks.size();
            for (i=0; i<size; i++) {
                tlink = (Link)(node.ContextLinks.elementAt(i));
                if (tlink != null) {
                    str = " dmp:ContextLink" + i + " "

```

05/23/00  
00:34:55

# Confidential and Proprietary Property of Rocky Nevin

GUI.java

35

```
c+"";

    "+" links nodes "+" +link.NodeR.Name+ "" and "" +link.NodeL.Nam
System.err.println(str);

    }
}

System.err.println("-----");

return;
} // end dump_node()

/**
 ** dump_nodes
 **
 */
public void dump_nodes (int debug_value, Node node) {
    int i, j, size;
    if (node == null)
        return;
    dump_nodes_recursive(debug_value, node);
} // end dump_nodes()

/**
 ** dump_nodes
 **
 */
public void dump_nodes_recursive (int debug_value, Node node) {
    int i, j, size;
    if (recursion_depth == 0)
        P(debug_value, "dump_nodes", "-----");
    dump_node(debug_value, false, node);
    recursion_depth++;

    size = node.Links.size();
    for (i=0; i<size; i++) {
        if (node.getNodeAllLink(i).hasLargerDistThan(node))
            dump_nodes(debug_value, node.getNodeAllLink(i));
    }

    recursion_depth--;
    if (recursion_depth == 0)
        P(debug_value, "dump_nodes", "-----");
    return;
} // end dump_nodes_recursive()

/**
 ** run1
 **
 */
*/
public void run1 () {
    update(1);
} // end run1

/**
 ** method update
 ** paint nodes, starting on Root
 */
public void update (int count) {
    int i;
    for (i=0; i<count; i++)
        update(true);
} // end update(int)

/**
 ** request.stop_thread block and wait for a limited while until Animating is false, return ultim
ely.
 **
 */
static public void request.stop_thread () {
    int i;

    StopThreadRequest = true; // sensed by ...

    for (i=0; (i<100 && Animating==true); i++) {
        System.err.println("");
        sleep(10);
    }
    return;
} // end request.stop_thread

/**
 ** method update
 ** update and paint nodes, starting on Root
 */
public void update (boolean change_images) {
    // change image_counter and get appropriate graphics in prep for paint()
    if (change_images) {
        if (image_counter>=2)
```

```
        {
            image_counter=1;
            graphics = image1.getGraphics();
        }
        else
        {
            image_counter=2;
            graphics = image2.getGraphics();
        }
    }

    // SET THE GRAPHICS TO THE IMAGE
    if (datasea.POV != null) {
        paint(datasea.POV, 0);
    }
    else {
        paint(datasea.Root, 0);
    }
    graphics.setColor(color_obj.LightGrey);

    // SET GRAPHICS TO THE FRAME'S GRAPHICS IN PREP FOR drawImage()
    graphics = frame.getGraphics();
    this.getToolkit().sync();
    // DRAW THE IMAGE ONTO THE FRAME
    if (image_counter==1)
        graphics.drawImage(image1, 0,0, frame);
    else
        graphics.drawImage(image2, 0,0, frame);

    if (auto_rescale)
        rescale("run");
    } // end update

/**
 ** method reset_current_TS , called by update
 */
public static void reset_current_TS () {
    current_TS = java.lang.System.currentTimeMillis();
} // end reset_current_TS

/**
 ** set_Xnode
 **
 */
public void set_Xnode () {

    int i, size;
    Node child;

    if (lastNode == Xnode) {
        Xnode = null;
        P(0,"set_Xnode ", "Setting Xnode to null");
    }
    else
        if (lastNode != null) {
            Xnode = lastNode;
            P(0,"set_Xnode ", "Setting Xnode to "+Xnode.Name);
        }
    else
        WARNING(0,"set_Xnode ", "Need a lastNode to set Xnode.");

    } // end set_Xnode

/**
 ** method paint , called by update
 */
public void paint (Node POV, int dist, to_POV) {
    float completion_level = 0;
    reset_current_TS();
    if (Debug==1)
        timer.start_timer("position");
    rescale("clear");
    if (Xnode != null) { // 11/27/99
        DataSea.needdistUpdate = true;
        completion_level = position_start(POV);
    }
    if (Xnode != null) { // 11/27/99
        DataSea.needdistUpdate = true;
        completion_level = position_start(Xnode);
    }
    if (Debug==1)
        timer.end_timer("position");
    if (Debug==1)
        timer.start_timer("render");
    completion_level = render_start(POV);
    if (Debug==1)
```

05/23/00  
00:34:55

## Confidential and Proprietary Property of Rocky Nevin

GUI.java

87

```
timer_end_timer("render");

) // end paint

/**
 ** get_angle
 ** set the angle theta of a node to angle from it's caller
 **
 */
// sic: sic public double get_angle(Node a, Node b) {
double delta_x, delta_y;
delta_x = b.x - a.x;
delta_y = b.y - a.y;
return (get_angle(delta_x, delta_y));
} // End of get_angle (Node...)

/**
 ** get_angle
 ** set the angle theta of a node to angle from it's caller
 **
 */
static public double get_angle(double delta_x, double delta_y) {
return(Math.atan2(delta_y,delta_x));
} // End of get_angle (double...)

/**
 ** action
 **
 */
public void action (Node node) {
int i, size, j, sizej;
Node child, grand_child;

if (node == null)
return;

if (node.isURL) {
text_frame.setVisible(true);
text_graphics = text_frame.getGraphics();
text_frame.setTitle(node.Name);
if (node.Data[0] != null) {
text_graphics.clearRect(0,0, (text_frame.getSize().width, text_frame.getSize().height);
draw_text(node);
}
}
else
if (node.isAN) {
text_frame.setTitle("Hidden");
text_frame.setVisible(false);
}
else
if (node.isDN) { // something like a menu
text_frame.setTitle("URL not selected");
text_frame.setVisible(false);
node.set_mag(Node.MAX_MAG);
datasea.back_r((Node)null, node, 0);
datasea.back_r((Node)null, node, 0);
datasea.back_r((Node)null, node, 0);
size = node.Links.size();

// make children all visible
for (i=0; i<size; i++) {
child = (Node)(node.getNodeAtLink(i));
if (child.dist > node.dist) {
child.set_mag(Node.MAX_MAG - 1);
sizej = child.Links.size();

// make grand-children's lines all visible, but no names
for (j=0; j<sizej; j++) {
grand_child = (Node)(child.getNodeAtLink(j));
if (grand_child.dist > child.dist) {
grand_child.set_mag(text.threshold - 0.1);
}
}
}
}
datasea.needdistUpdate = true;

} // end action

/**
 ** draw_text
 **
 */
```

05/23/00  
00:34:55

# Confidential and Proprietary Property of Rocky Nevin GUI.java

38

```
*/
public void draw_text (Node node) {
    int i, size;
    String s=null;

    // TEST HERE

    size = Node.MAX_TEXT_DATA_LINES;
    for (i=0; i<size; i++) {
        s = node.Data(i);
        if (s == null)
            break;
        else {
            s = eliminate_html(s);
            text_graphics.drawString(s, 10,50+10*i);
        }
    } // end draw_text

}

/**
 ** find_domain_name
 **
 */
public String find_domain_name (String input_string) {
    int index, num_words;
    String words[], ret_string=null;

    if (-1 == (index = input_string.toLowerCase().indexOf("http://")))
        return((String)null);

    Sys.out.println("find_domain_name: full string is "+input_string);

    StringTokenizer st = new StringTokenizer(input_string, "/");
    num_words = st.countTokens();
    words = new String[num_words];

    if (num_words >= 2) {
        ret_string = st.nextToken();
        System.out.println("find_domain_name: 'http:' is ===== "+ret_string);
        ret_string = st.nextToken(); // use the second one
        System.out.println("find_domain_name: Assuming domain is ===== "+ret_string);
    }

    /*****
    for (int i = 0; i < num_words; i++) {
        words[i] = st.nextToken();
        index = words[i].toLowerCase().indexOf("www.");
        if (0 <= index)
            ret_string = words[i];
    }
    *****/
    return(ret_string);
} // end find_domain_name

/**
 ** find_URL_name
 **
 */
public String find_URL_name (String input_string) {
    int index, num_words;
    String ret_string=null;
    String words[];

    if (-1 == (index = input_string.toLowerCase().indexOf("href=")))
        return((String)null);

    StringTokenizer st = new StringTokenizer(input_string, "<>\"'\\n");
    num_words = st.countTokens();
    words = new String[num_words];
    //System.out.println("find_URL_name: full string is --> "+input_string+"<--", num_words="+num_w
ords);

    for (int i = 0; i < num_words; i++) {
        words[i] = st.nextToken();
        //System.out.println("find_URL_name: Working on ===== "+words[i]);
        index = words[i].toLowerCase().indexOf("href");
        if (0 <= index) { // found one
            if ((i+1)<num_words) {
                words[i++] = st.nextToken();
                ret_string = words[i];
            }
            else
                System.out.println("find_URL_name ERROR, FOUND 'HREF' but i+1 >= nu
m_words");
        }
    }

    // See if there are bad things here ...
```

05/23/00  
00:34:55

GUI.java

99

```
if (ret_string != null) {
    if (0 <= (index = ret_string.toLowerCase().indexOf("mailto")))
        ret_string = null;
    else
        if (0 <= (index = ret_string.toLowerCase().indexOf("cgi-bin")))
            ret_string = null;
        else
            if (0 <= (index = ret_string.toLowerCase().indexOf("#")))
                ret_string = null;
            else
                if (0 <= (index = ret_string.toLowerCase().indexOf("messages"))) // for egg-tempera
                    ret_string = null;
                }
            //if (ret_string != null)
            //    System.out.println("find_URL.name returning "+ret_string);

    return(ret_string);
} // end find_URL_name

/**
 ** eliminate.html eliminate what look like HTML tags, replace with a blank
 */
public String eliminate_html (String string_arg) {
    String s=string_arg;
    int left_bracket_index, right_bracket_index;
    String w1, w2;

    while (true) {
        if (s.length() <= 0) {
            break;
        }
        if (0 <= s.toLowerCase().indexOf("meta")) {
            break;
        }
        left_bracket_index = s.indexOf("<");
        right_bracket_index = s.indexOf(">");
        if (left_bracket_index < 0)
            break;
        if (right_bracket_index < 0)
            break;
        if (right_bracket_index <= left_bracket_index) {
            //System.err.println("draw text breaking:right_bracket_index <= left_bracket_in
            //System.err.println("draw text breaking:right_bracket_index <= left_bracket_in
        }
    }
}
```

```
dex");
    break;
}
w1 = s.substring(0, left_bracket_index);
w2 = s.substring(right_bracket_index+1);
s = w1 + " " + w2;
}
return(s);
} // end eliminate_html

/**
 ** eliminate.punctuation eliminate what look like HTML tags, replace with a blank
 */
public String eliminate_punctuation (String string_arg) {
    String s=string_arg;
    int left_bracket_index, right_bracket_index;
    String w1, w2;

    while (true) {
        if (s.length() <= 0) {
            break;
        }
        if (0 <= s.toLowerCase().indexOf("meta")) {
            break;
        }
        left_bracket_index = s.indexOf("<");
        right_bracket_index = s.indexOf(">");
        if (left_bracket_index < 0)
            break;
        if (right_bracket_index < 0)
            break;
        if (right_bracket_index <= left_bracket_index) {
            //System.err.println("draw text breaking:right_bracket_index <= left_bracket_in
            //System.err.println("draw text breaking:right_bracket_index <= left_bracket_in
        }
        w1 = s.substring(0, left_bracket_index);
        w2 = s.substring(right_bracket_index+1);
        s = w1 + " " + w2;
    }
    return(s);
} // end eliminate_punctuation
```

05/23/00  
00:34:55

Confidential and Proprietary Property of Rocky Nevin  
GUI.java

410

```
/**
 **  p  print out something
 **
 */
public void p (String s) {
    System.out.println(s);
} // end p

/**
 **  node.to_string
 **
 */
public String node.to_string (Node node) {
    String s = "<"+node.Name+">(m="+node.mag+")("+node.Type+")[d="+node.dist+"]Td="+node.
    Tdist+"]";
    return(s);
} // end node.to_string

} // End of Object GUI
```

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin

### DataSea.java

// This is DataSea.java by Rocky Nevin

```
import java.applet.*;
import java.lang.*;
import java.awt.*;
import java.util.*;
import java.io.*;
```

/\*\*

\* This is DataSea.java by Rocky Nevin

\* @version 0.1, 8/6/98

\* @version 0.4, 3/12/98

\*

\* 8/6/98

\* New version, rewritten

\* Ported to JBuilder2 9/98

\* Principal Methods:

absorb\_POV

add\_POV

add\_to\_node\_vec

add\_to\_unprocessed\_vec

closer

connect\_up

find\_node\*

gen\_\*

init

locate\_node

more\_attraction/repulsion

populate

PostProcessor

show\_node\_vec

spread\_mag

string\_input

world\_\* methods parsing input like 'show', 'back', 'more', 'sim' ...

These all used to be word\_xxx or word\_xxx\_start, but many now are simply xxx, like 'word.back.start' is now 'start'.

1/19/99

Node-specific rendering is important and must be included. Nodes with master-rendering function will invoke their children's subservient rendering.

1/20/99  
Add potential function to the spread mechanism: potentiate 1+ nodes which spreads, and then stimulate 1+ other nodes. Any nodes both

stimulated and potentiated change their magnitude, others that are only stimulated are not changed, or at least not as much.

4/15/99

Add Group(int level) which will go from POV to dist=level and from there force all children of dist=(level-1) to position themselves on parent (dist=level), and travel back up proximally to POV. Subsequent invocations on (level-1) will group things reasonably.

\*/

public class DataSea extends Object {

static GUI gui; // The GUI passed to the DataSea constructor, which created us  
static Node currentNode=null; // to link POV to easily

static Node CN; // to link POV to easily

static Node Notes; // to link POV to easily

static Node Myself; // This is a self-node, one which can be used to show the DataSea

// program itself

static Node Root;

static String useless\_words; // exclude these ... make exclusion list. discard

static Node Thesaurus\_node;

static Node Populate\_pop;

// static Node BG;

static Node POV = null;

static Vector node\_vec; // holds all nodes for searching later

static Vector list\_vec; // holds nodes for passing between routines

static Vector big\_mag\_node\_vec; // holds nodes for passing between routines

static int GlobalDistalCount = 0;

static int event\_counter = 0;

Vector GlobalVec = null;

Vector unprocessed\_vec;

// int maxnodes=1000;

static int link\_ID = 0;

int POV\_name = 0;

static boolean needdistUpdate=false;

static boolean theta\_org=true;

static double delta\_dist = 0.1; // used to increment Node.dist when types are same

boolean whats = false;

public DataSea (Object gui\_argument) { // Constructor  
gui = (GUI)gui\_argument; // Now we can refer to our creator  
this.init();  
} // end DataSea constructor

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin

DataSea.java

2

```
public void init () {
    pop = new Populate(this);
    init_useless_words();
    if (Myself == null) {
    }
    return;
}

/**
 ** init_useless_words
 **
 */
public void init_useless_words () {
    int i, size;
    Node child;

    String w;
    w = "a an any are as at be by fix have has himl if is rel see the to me not on pdt public ";
    w = w+"this written about how's going would be it";
    w = w+"said says with you don't forget close please your";
    w = w+"for next turn dear named my day happy input";
    w = w+"interesting believes of most";
    w = w+"yesterday today tomorrow lastweek thisweek";
    w = w+"---< > = & [ \\. : ; , ";
    w = w+"Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec";
    w = w+"1999 2000";
    w = w+"1 2 3 4 5 6 7 8 9 10 11 12 13 14 15";
    w = w+"16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31";
    w = w+"";

    // Ignore case
    useless_words = w.toLowerCase();

    //System.out.println("Useless_words: "+useless_words);
    return;

} // end init_useless_words

/**
 ** reset_node_IDs
 **
 */
public void reset_node_IDs () {
    int size, i;

    Node node, child;

    size = node.vec.size();
    for (i=0; i<size; i++) {
        node = (Node)(node.vec.elementAt(i));
        node.ID = 0;
    }
    // end reset_node_IDs

    /**
     ** cats Lift mag of downstream nodes of same type.
     ** If different type, lift half and quit
     **
     */
    public void cats (String words[], int num_words) {
        int i, size;
        Node node, child;

        if (null == (node = figure_out_node("cats", words, num_words)))
            return;

        reset_node_IDs();

        node.lift(2); // make sure the starting point isn't left in the dust

        size = node.Links.size();
        for (i=0; i<size; i++) {
            child = node.getNodeAtLink(i);
            if (child.hasLargerDistThan(node)) {
                System.out.println("cats "+child.Name);
                cats(node, child, i+1);
            }
        }
    } // end cats

    /**
     ** cats Show categories (A.N's) from node by calling new_back_r on AN's hit > once
     **
     */
    public void cats (Node caller, Node node, int id) {
        int i, size;
        Node child;

        if (node==null)
    }
```

```

    else {
        child.lift(2);
        System.out.println("magdownstream halting on "+child.Name);
    }
}

} // end magdownstream

/**
 ** magdownstream
 **
 */
public void magdownstream (Node node) {
    int i, size;
    Node child;

    if (node==null)
        return;

    size = node.Links.size();
    for (i=0; i<size; i++) {
        child = node.getNodalLink(i);
        if (child.hasLargerDistThan(node))
            if (node.goestDownstreamTo(child) && node.Type==child.Type) {
                child.lift(2);
                System.out.println("magdownstream "+child.Name);
                magdownstream(child);
            }
        else {
            child.lift(1);
            System.out.println("magdownstream halting on "+child.Name);
        }
    }
}

} // end magdownstream

/**
 ** method backs
 */
amplify mag going backwards, spread to first children also. Calls new_back_r
/

```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

DataSea.java

4

```
public void backs (String[] words, int num_words) {
    int i, size;
    Node node, tnode;

    if (num_words<1)
        return;

// HANDLE IDENTIFYING THE CORRECT NODE TO START ON
    if (num_words==1)
        node = gui.lastNode;
    else
        node = find_node.named(words[1]);

    if (words[0].equalsIgnoreCase("whats")) {
        whats = true;
    }

// CHECK FOR ERRORS
    if (node==null) {
        if (num_words>1)
            GUI.WARNING(0,"DataSea.backs","Can't find node "+words[1]);
        else
            GUI.WARNING(0,"DataSea.backs","Neither Name given nor existing lastNode.");
        return;
    }
    else
        GUI.P(1,"DataSea.backs","Found node named '"+node.Name+"'");
    backs(node);
}

/**
 ** method backs  amplify mag going backwards, spread to first children also. Calls new_back_r
 */
public void backs (Node node) {
    int i, size;

    boolean Saved_StopSpread = node.StopSpread; // Temporarily change it
    node.StopSpread = true;

    GUI.SavedNode = node; // used by 'why' later

//System.out.println("backs(): new_back_r("+node.Name+")");
//new_back_r((Node)null, node, 0, false);

Node tn=null;
// ----- for children -----
size = node.Links.size();
for (i=0; i<size; i++) {
    tn = (Node)(node.getNodalLink(i));
    if (tn.dist != -1) { // That is, there is a path to POV from here ...
        //System.out.println("");
        //System.out.println("backs(): new_back_r("+tn.Name+")");
        new_back_r((Node)tn, 0, false);
    }
}
// ----- end for children -----

node.StopSpread = Saved_StopSpread; // Restore it
return;
} // end backs

/**
 ** method back  amplify mag going backwards, calls back_r
 */
public void back (String[] words, int num_words) {
    int i, size;
    Node node, tnode;

    if (null == (node = figure_out_node("back", words, num_words)))
        return;

    if (words[0].equalsIgnoreCase("whats")) {
        whats = true;
    }

// NOW, MAKE RECURSIVE CALL
set_Tdist_start(POV); // NEED FOR SETTING VARIABLE 'TDIST'
back_r((Node)null, node, 0);
return;
} // end back

/**
 ** method new_back_r  amplify mag going backwards
 */
public double new_back_r (Node caller, Node child, int current transition count, boolean previous)
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

## DataSea.java

5

```
y.entered_AN) { // returns product of accumulated CS's
int i, size, passed_transition_count=current_transition_count;
double caller_mag, ret_value=0.0, recursion_value=1.0;
Node grand_child;
String caller_name = "(caller_name is null)";
char pol1, pol2; // the polarization symbols for links
boolean entered_AN=false; // we can start on an AN and then exit a string of them, tha
t's OX.

if (cl_hld == null) // so we need to see if we both enter and exit ANs
    return(0.0);
if (child == Root)
    return(0.0);

String Cn = " caller=" + gui.node.to_string(caller);
String cn = " child=" + gui.node.to_string(child);
String gcn:

if (child_dist <= 2) { // stop near POV
    GUI.P(1, "back_r", "OK, stopping near POV at child="+cn);
    return(1);
}

if (current_transition_count > GUI.max_transition_count) {
    -----
if (caller.Debug || child.Debug)
    gui.p("nbr: current_transition_count > GUI.max_transition_count: "+Cn+cn);
    return(0); // Need to return 0 which gets multiplied by prior recursion values ...
    // want this thread to go backwards with the message 'don't magnify'
}

if (!caller.isAN && child.isAN) // Sense going into an AN from a non-AN
    previously_entered_AN=true;

//HERE

size = child.Links.size();
for (i=0; i<size; i++) {
    grand_child = child.getNodeAtLink(i);
    gcn = " grand_child=" + gui.node.to_string(grand_child);
    if (grand_child==POV)
        {return(1.0);}
    if (grand_child==caller)
        (continue;)
    if (grand_child_dist>-1
        && !grand_child.hasArcerDistThan(child)) // Here WE CHECK ORDER OF

NODES
{
    entered_AN=previously_entered_AN; // reset
    if (!child.isAN && grand_child.isAN) // Sense going into an AN from a non-AN
        {
            ----- Entering AN
            entered_AN=true;
            if (grand_child.Debug) gui.p("nbr: Entering grand_child AN: "+Cn+cn+gcg);
        }
        if (previously_entered_AN && !grand_child.isAN) // Exiting, don't pursue into and out of an AN
            ----- Exiting AN
            {
                if (child.Debug || grand_child.Debug)
                    gui.p("nbr: Exiting AN, previously_entered_AN=true, g-child!=AN "+Cn+cn
                    +gcg);
                continue;
            }
            // RECURSION Here
            pol2 = child.getPol(grand_child);
            if (pol2!='+') {
                ----- Increment tran-count, recur
                se
                if (grand_child.Debug || child.Debug) gui.p("nbr: + pol and recursing to grand_c
                hild: <"+Cn+cn+gcg);
                recursion_value = new_back_r(child, grand_child, 1+passed_transition_count, ent
                ered_AN); // recurse on any, check Type in beginning
                else {
                    ----- Just recurse
                    if (grand_child.Debug || child.Debug) gui.p("nbr: not + pol but recursing to gra
                    nd_child: <"+Cn+cn+gcg);
                    recursion_value = new_back_r(child, grand_child, passed_transition_count, enter
                    ed_AN); // recurse on any, check Type in beginning
                }
                if (recursion_value > ret_value) { // Notice success reaching back home
                    if (grand_child.Debug || child.Debug) gui.p("nbr: Success sensed dist
                    al to: <"+Cn+cn+gcg);
                    ret_value = recursion_value;
                }
            }
            ret_value *= child.potential_mag;
            if (ret_value > 0.0) {
                -----
                if (caller.Debug || child.Debug) gui.p("nbr: mag'ing child: "+Cn+cn+cn);
                child.set mag(ret_value * Node.EMPHASIZED_MAG);
            }
        }
    }
}
```

05/23/00  
00:48:20

Confidential and Proprietary Property of Rocky Nevin  
DataSea.java

6

```
    }
    return(ret_value);
} // end new_back_r

/**
** method back_r amplify mag going backwards, // 6/7/99 recurse if type==AN
*/
public double back_r (Node caller, Node child, int current_transition_count) { // returns product of a
    accumulated CS's
    int i, size, passed_transition_count=current_transition_count;
    double caller_mag, ret_value=0.0, recursion_value=1.0;
    Node grand_child;
    String caller_name = "(caller_name is null)";

    if (child == null)
        return(0.0);

    if (child.dist <= 1) { // stop near POV
        return(1);
    }

    size = child.links.size();
    for (i=0; i<size; i++) {
        grand_child = child.getNodeAtLink(i);
        if (grand_child==POV)
            return(1.0);
        if (grand_child==null) {
            if (grand_child.dist>-1
                && grand_child.hasSmallerDistThan(child)) // Here WE CHECK ORDER O
                FNODES
            {
                // RECURSION Here
                recursion_value = back_r(child, grand_child, passed_transition_count); // recurse
                on any, check Type in beginning
                if (recursion_value > 0)
                    if (recursion_value > ret_value) {
                        ret_value = recursion_value;
                    }
            }
        }
    }

}

}

if (caller!=null)
    caller_name = caller.Name;
ret_value *= child.potential_mag;
if (ret_value > 0.0) {
    child.set_mag(ret_value * Node.EMPHASIZED_MAG);
}
return(ret_value);
} // end back_r

/**
** method threshold_threshold_back_r amplify mag going backwards, // 6/7/99 recurse if type==
=AN
*/
public double threshold_back_r (Node caller, Node child) { // returns product of accumulated CS's
    int i, size;
    double caller_mag, ret_value=0.0, recursion_value=1.0;
    Node grand_child;
    String caller_name = "(caller_name is null)";

    if (child == null)
        return(0.0);

    if (child.dist <= 1) { // stop near POV
        return(1);
    }

    child.lift_to_threshold();
    size = child.links.size();
    for (i=0; i<size; i++) {
        grand_child = child.getNodeAtLink(i);
        if (grand_child==POV)
            return(1.0);
        if (grand_child==null) {
            if (grand_child.dist>-1
                && grand_child.hasSmallerDistThan(child)) // Here WE CHECK ORDER O
                FNODES
            {
                // RECURSION Here
                recursion_value = threshold_back_r(child, grand_child) // recurse on any, check
                Type in beginning
            }
        }
    }

}
```

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin

DataSea.java

7

```
    }
    return(ret.value);
} // end threshold_back_r

}

/**
 ** remember set Node.min_mag to the current mag (generally, don't allow it to shrink)
 **
 */
public void remember () {
    int i, size;
    Node node=null;
    Node child=null;

    if (gui.lastNode == null)
        return;

    GUI.P(0,"remember","lastNode="+gui.lastNode);
    GUI.P(0,"remember","lastNode.Name="+gui.lastNode.Name);
    GUI.P(0,"remember","lastNode.Links.size()="+gui.lastNode.Links.size());

    size = gui.lastNode.Links.size();
    for (i=0; i<size; i++) {
        child = (Node)(gui.lastNode.getNodAAtLink(i));
        //if (child.dist == gui.lastNode.dist+1)
            if (child.dist > gui.lastNode.dist)
                remember_r(gui.lastNode, child);
    }

} // end remember

/**
 ** go_until_AN_DN
 **
 */
public void go_until_AN_DN (Node node, boolean did_trans_to_AN) {
    int i, size;
    Node child;
    boolean is_trans_to_AN=false, is_trans_to_DN=false;

    GUI.P(0,"go_until_AN_DN", node.Name+" did_trans_to_AN="+did_trans_to_AN);
    size = node.Links.size();
    for (i=0; i<size; i++) {
        child = node.getNodAAtLink(i);
        if (((child.dist == node.dist+delta.dist) || (child.dist == (int)(node.dist+1))
            {
                if (!node.isAN && child.isAN)
                    is_trans_to_AN = true;
                if (node.isAN && !child.isAN)
                    is_trans_to_DN = true;
                if (did_trans_to_AN && is_trans_to_DN) {
                    GUI.P(0,"go_until_AN_DN", child.Name+" halted: trans_to_DN=true");
                    break;
                }
                go_until_AN_DN(child, (is_trans_to_AN || did_trans_to_AN));
            }
        }

    } // end go_until_AN_DN

    /**
     ** like
     **
     */
    public void like (Node node) {
        go_until_DN_AN((Node)null, node, '+');
        go_until_DN_AN((Node)null, node, '-');
    } // end like

    /**
     ** go_until_DN_AN
     **
     */
    public void go_until_DN_AN (Node caller, Node node, char sign) {
        int i, size, dist_diff;
        Node child;
        boolean is_trans_to_AN=false;
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

DataSea.java

8

```
if (node == null)
    return;

GUI.P(0,"go_until_DN_AN", "direction "+sign+", "+node.Name);
if (caller!=null)
    caller.more_CS(node);

if (sign=='+')
    dist_diff = 1;
else
    dist_diff = -1;

size = node.Links.size();
for (i=0; i<size; i++) {
    child = node.getNodeAtLink(i);
    if ((child.dist == (node.dist+dist_diff) && child.dist > 2) {
        if (!node.isAN && child.isAN)
            is_trans_to_AN = true;
        if (is_trans_to_AN) {
            GUI.P(0,"go_until_DN_AN", child.Name+", halted: trans_to_AN=true");
            break;
        }
        go_until_DN_AN(node, child, sign);
    }
}

return;
} // end go_until_DN_AN

/**
 ** addCNodeBetweenNodes : DON'T USE: add a modulating CNode to the link between node1
 ** and node2
 */
public void addCNodeBetweenNodes (Node node1, Node node2, Node CNode) { // CSs

    Link link = node1.getLinkTo(node2);
    link.addCNode(CNode);
} // end addCNodeBetweenNodes

/**
 ** remember_r set CS between parent and node based on current mag
 **
 */
public void remember_r (Node parent, Node node) {
    int i, size;
    Node child=null;
    double cs, old_cs;

    if (parent==null)
        return;
    if (node==null)
        return;

    cs = parent.get_CS(node);
    old_cs = cs;

    Link link = parent.getLinkTo(node);
    if (link == null) {
        gui.ERROR(0, "remember_r", "link is null to "+node.Name);
        return;
    }

    if (node.mag >= Node.MAX_MAG)
        cs *= 1.4;
    else if (node.mag >= Node.BIG_MAG)
        cs *= 1.2;
    else if (node.mag >= Node.DEFAULT_MAG)
        ;
    else
        cs /= 1.2;

    link.set_CS(cs);
    GUI.P(0,"remember_r",node.Name+", CS changing from "+gui.prec(old_cs,3)+" -> "+gui.prec(cs,3));
}

size = node.Links.size();
for (i=0; i<size; i++) {
    child = node.getNodeAtLink(i);
    //if (child.dist == node.dist+1)
    if ((child.dist == node.dist+delta_dist) || (child.dist == (int)(node.dist+1)))
        remember_r(node, child);
}

} // end remember_r
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

DataSea.java

9

```
/**
 ** forget set Node.min_mag to the current mag (generally, don't allow it to shrink)
 **
 */
public void forget () {
    int i, size;
    Node node;
    Node child;

    size = node.vec.size();
    for (i=0; i<size; i++) {
        node = (Node)(node.vec.elementAt(i));
        node.min_mag = node.mag;
    }

} // end forget

/**
 ** showevents
 **
 */
public void showevents () {
    int i, size;
    Node node;
    Node child;

    size = node.vec.size();
    for (i=0; i<size; i++) {
        node = (Node)(node.vec.elementAt(i));
        if (node.isEvent) {
            if (null != node.hasSiblingOfType("Event")) {
                if (node.mag < Node.BIG_MAG)
                    node.set_mag(Node.BIG_MAG);
            }
            else
                node.set_mag(Node.MAX_MAG);
        }
    }
}

return;
```

```
} // end showevents

/**
 ** showtime
 **
 */
public void showtime () {
    int i, size;
    Node node;
    Node child;

    mag_r((Node)null, pop.Timeline, "distal", 3, 0, "+", false);

    /*****
    size = node.vec.size();
    for (i=0; i<size; i++) {
        node = (Node)(node.vec.elementAt(i));
        if (node.isEvent) {
            if (node.mag < Node.MAX_MAG)
                node.set_mag(Node.MAX_MAG);
        }
    }
    *****/

    return;
} // end showtime

/**
 ** peg set Node.min_mag to the current mag (generally, don't allow it to shrink)
 **
 */
public void peg () {
    int i, size;
    Node node;
    Node child;

    size = node.vec.size();
    for (i=0; i<size; i++) {
        node = (Node)(node.vec.elementAt(i));
        node.min_mag = node.mag;
    }
}

} // end peg
```

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin

### DataSea.java

10

```
/*
 * find_AN_named
 */
public Node find_AN_named (String name)
{
    int i, size;
    Node tn=null, ret_node = null;

    size = node_vec.size();
    GUI.P(2, "find_AN_named", "size="+size+" name="+name);
    for (i=0; i<size; i++) {
        tn = (Node)(node_vec.elementAt(i));
        if (tn.Name.equalsIgnoreCase(name) && tn.isAN()) {
            GUI.P(2, "DS.find_AN_named", "Found "+name);
            ret_node = tn;
        }
    }
    if (ret_node == null)
        GUI.ERROR(2, "DS.find_AN_named", "Null node for name "+name);
    else
        GUI.P(2, "DS.find_AN_named", "FOUND node for name "+name);

    return(ret_node);
} // end find_AN_named

/*
 * find_DN_named
 */
public Node find_DN_named (String name)
{
    int i, size;
    Node tn=null, ret_node = null;

    size = node_vec.size();
    GUI.P(2, "find_DN_named", "size="+size+" name="+name);
    for (i=0; i<size; i++) {
        tn = (Node)(node_vec.elementAt(i));
        if (tn.Name.equalsIgnoreCase(name) && tn.Type.equalsIgnoreCase("DN")) {
            GUI.P(2, "DS.find_DN_named", "Found "+name);
            ret_node = tn;
        }
    }
}

if (ret_node == null)
    GUI.ERROR(2, "DS.find_DN_named", "Null node for name "+name);
else
    GUI.P(2, "DS.find_DN_named", "FOUND node for name "+name);

return(ret_node);
} // end find_DN_named

/**
 ** moreless   moreless adjust mags to see structure of nodes better, count num visible
 **
 */
public void moreless (String cmd) {
    Node node;

    int count=0;
    for (int i=0; i<node_vec.size(); i++) { // count them
        if ((node==(Node)(node_vec.elementAt(i))) != null)
            if (node.mag > gui.text.threshold)
                count ++;
    }

    if (gui.desired_visible_count == 0) {
        gui.desired_visible_count = count;
        GUI.P(0, "moreless", "desired_visible_count initially set to "+gui.desired_visible_count);
    }

    if (cmd.equals("m")) {
        gui.desired_visible_count *= 1.5;
        GUI.P(0, "moreless", "desired_visible_count up to "+gui.desired_visible_count);
        auto_flatten();
    }
    else {
        gui.desired_visible_count /= 1.5;
        if (gui.desired_visible_count < 5)
            gui.desired_visible_count = 5;
        GUI.P(0, "moreless", "desired_visible_count down to "+gui.desired_visible_count);
        auto_flatten();
    }
}

return;
} // end moreless

/**
 ** auto flatten   automatically adjust mags to see structure of nodes better. count num visible
 */
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin DataSea.java

111

```

**
*/
public void auto_flatten () {
    int i, size, count=1000000, iteration_count = 0;
    Node tn, node;

    while ((count > (gui.desired_visible_count+5)) && (iteration_count++ < 100)) { // impose a limit of
    100 tries
        count = 0;
        for (i=0; i<node_vec.size(); i++) { // count them
            if ((node==(Node)(node_vec.elementAt(i))) != null)
                if (node.mag > gui.pos_threshold)
                    count++;
        }
        GUI.P(0, "DataSea.auto_flatten first-half", iteration_count+" "+ "+count+" nodes > threshold, w
        ant "+gui.desired_visible_count);

        if (count > gui.desired_visible_count) { // raise threshold
            sharpen();
        }

        System.err.println("count="+count+", gui.desired_visible_count = "+gui.desired_visible_count);

        // count and decrease the threshold if needed
        iteration_count = 0;
        while ((count < (gui.desired_visible_count) && (iteration_count++ < 100)) { // impose a limit of 1
        00 tries
            count = 0;
            for (i=0; i<node_vec.size(); i++) { // count them
                if ((node==(Node)(node_vec.elementAt(i))) != null)
                    if (node.mag > gui.pos_threshold)
                        count++;
            }
            GUI.P(0, "DataSea.auto_flatten second-half", iteration_count+"") -- -- "+count+" nodes < threshold,
            want "+gui.desired_visible_count);

            // adjust the threshold
            if (count < gui.desired_visible_count) { // raise threshold
                flatten();
            }
        }
        return;
    } // end auto_flatten
}

/**
** DataSea.show_node_vec
**
*/
public int show_node_vec () {
    int i;
    for (i=0; i<node_vec.size(); i++)
        if (node_vec.elementAt(i) != null)
            GUI.P(1, "DataSea.show_node_vec", (((Node)node_vec.elementAt(i)).Name));
    return(0);
}

/**
** DataSea.add_to_node_vec
**
*/
public static int add_to_node_vec (Node dn) {
    if (node_vec == null)
        node_vec = new Vector(100);
    node_vec.addElement(dn);
    // MAYBE NOT A GOOD IDEA TO LINK ROOT TO EVERYTHING 3/24/99
    // if (Root != null)
        Root.link(dn);
    return(0);
}

/**
** DataSea.clear_list clear the vector used to pass groups of nodes
**
*/
public static void clear_list () {
    if (list_vec == null)
        list_vec = new Vector(10);
    else
        list_vec.removeAllElements();
    return;
} // end clear_list

/**
** DataSea.add_to_list add to the vector used to pass groups of nodes
**

```

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin

DataSea.java

12

```
*/
    public static void add_to_list (Node node) {
        if (list_vec == null)
            list_vec = new Vector(10);
        list_vec.addElement(node);
    }
    return;
} // end add_to_list

/**
 ** DataSea.stored_into_list_already clear the vector used to pass groups of nodes
 **/
    public static boolean stored_into_list_already (Node node) {
        if (!list_vec.contains(node))
            return(true);
        else
            return(false);
    } // end stored_into_list_already

/**
 ** cl 'create_and_link'
 **/
    public Node cl (String Name) {
        Node tnode;
        tnode = find_node_named(Name);
        if (tnode == null)
            tnode = cl(Name, "DN", "", Link.DEFAULT_CS);
        else
            this.link(tnode);
        return(tnode);
    }

    public Node cl (String Name, double cs) {
        Node tnode;
        // tnode = cl(Name, "DN", "", cs);
        tnode = DataSea.find_node_named(Name);
    }

    if (tnode == null)
        tnode = cl(Name, "DN", "", cs);
        return(tnode);
    }

    public Node cl (String Name, String Type1, String Name2, String Type2) {
        Node node1, node2, tnode;
        node1 = pop.create_node(Name1, Type1);
        node2 = pop.create_node(Name2, Type2);
        node1.link(node2);
        this.link(node1);
        return(node2);
    }

    // new Node sets CS if available into Node.link() sets it into proper CS_R or CS_L
    public Node cl (String Name, String Type, String Desc, double CS) {
        Node tnode;
        tnode = pop.create_node(Name, Type, Desc);
        this.link(tnode, Desc, CS);
        return(tnode);
    }

    public Node cl (String Name, String Type, String Desc,
        int x, int y) {
        Node tnode;
        tnode = cl(Name, Type, Desc, Link.DEFAULT_CS);
        tnode.X = x;
        tnode.Y = y;
        return(tnode);
    }
}

/*****
```

```

Node reIdn;
reIdn = cloner((Node) dn, "NoName");
return(reIdn);
}

public Node cloner (String name) {
    Node reIdn;
    reIdn = cloner((Node) null, name);
    return(reIdn);
}

public Node cloner () {
    Node reIdn;
    reIdn = cloner((Node) null, "NoName");
    return(reIdn);
}

/*
*/

/*
* find_node_named (overloaded) First look for AN, then any if not found
*/
static public Node find_node_named (String name)
{
    int i, size;
    Node tn=null, ret_node = null;

    size = node_vec.size();

    // Check for AN first, then anything
    if ((ret_node=find_node_named(name, "AN")) == null)
        ret_node=find_node_named(name, (String)null);

    return(ret_node);
} // end find_node_named

/*
*/
* find node named name, with Type of type
*/
static public Node find_node_named (String name, String type)
{
    int i, size;
    Node tn=null, ret_node = null;

    size = node_vec.size();

    for (i=0; i<size; i++) {
        tn = (Node)node_vec.elementAt(i);

```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

DataSea.java

14

```
if (type == null) {
    if (tn.Name.equalsIgnoreCase(name))
        ret_node = tn;
    } else {
        if (tn.Name.equalsIgnoreCase(name) && tn.Type.equalsIgnoreCase(type))
            ret_node = tn;
    }
}
return(ret_node);
} // end find_node_named

/**
 ** create_note see if it exists, create DN if not, return it
 ** Automatically create a matching AN for a DN
 */
public Node create_note (String Desc) {
    int i;
    Node inode=null;

    GUI.P(0,"create_note", "Desc="+Desc);
    inode = new Node("Note", "DN", Desc);
    return(inode);
} // end create_note

/**
 ** normalize
 **
 */
static public double normalize () {
    int i, size;
    Node in;

    Node max_mag_node=null;
    double max_mag=0, normalization_factor=1, delta;

    if ((GUI.doNormalization) {
        GUI.P(1,"normalize", "normalization off");
        return(max_mag);
    }

    // ----- for all nodes -----
    size = node_vec.size();
    for (i=0; i<size; i++) {
        in = (Node)(node_vec.elementAt(i));

        if (tn.mag > 11)
            tn.mag = 10+Math.sqrt(tn.mag-10.0);
        if (tn.mag > max_mag) {
            max_mag = tn.mag;
            max_mag_node = tn;
        }
    }
    // ----- end for all nodes -----

    if (max_mag < 10.0) {
        GUI.P(1,"normalize", "max_mag_node.Name+ ", max_mag="+max_mag+", normalization_factor="+normalization_factor);
        return(max_mag);
    }
    normalization_factor = 10.0 * 1.0/max_mag;

    GUI.P(1,"normalize", "max_mag_node.Name+ ", max_mag="+max_mag+", normalization_factor="+normalization_factor);

    // ----- for all nodes -----
    for (i=0; i<size; i++) {
        in = (Node)(node_vec.elementAt(i));
        in.set_mag_no_history(tn.mag * normalization_factor);
    }
    // ----- end for all nodes -----

    /*****
    System.out.println("normalize: log(max_mag)="+Math.log(max_mag));
    System.out.println("normalize: log(normalization_factor)="+Math.log(normalization_factor));
    System.out.println("normalize: log(1)="+Math.log(1));
    System.out.println("normalize: exp(0)="+Math.exp(0));
    System.out.println("normalize: exp(1)="+Math.exp(1));
    System.out.println("normalize: log(1.7)="+Math.log(1.7));
    System.out.println("normalize: log(3.7)="+Math.log(3.7));
    System.out.println("normalize: log(3.7)="+Math.log(3.7));
    *****/

    if (POV != null) {
        // ----- for children -----
        size = POV.Links.size();
        for (i=0; i<size; i++) {
            tn = (Node)(POV.getNodeAtLink(i));
            if (tn.mag < Node.BIG_MAG)
                tn.set_mag(Node.BIG_MAG);
        }
    }
}
```

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin

DataSea.java

15

```
// ----- end for children -----
    }

    return(max_mag);
} // end normalize

/**
 ** flatten add constant to everything, and then after normalization, all's flatter
 **/
public double flatten () {
    int i, size;
    Node tn;

    double max_mag=0.0, delta_mag=0.0, current_mag=0.0, new_mag=0.0;

    size = node_vec.size();
    // ----- for all nodes -----
    for (i=0; i<size; i++) {
        tn = (Node)(node_vec.elementAt(i));
        current_mag = tn.mag;
        delta_mag = (Node.MAX_MAG - current_mag)/10;
        new_mag = current_mag + delta_mag;
        tn.set_mag_no_history(new_mag);
    }
    // ----- end for all nodes -----

    GUI.P(0,"flatten", "Done.");
    return(max_mag);
} // end flatten

/**
 ** sharpen subtract constant to everything, and then after normalization, all's less flat
 **/
public double sharpen () {
    int i, size;
    Node tn;

    double max_mag=0.0, delta_mag=0.0, current_mag=0.0, new_mag=0.0;

    size = node_vec.size();
    // ----- for all nodes -----
    for (i=0; i<size; i++) {
        tn = (Node)(node_vec.elementAt(i));

        current_mag = tn.mag;
        delta_mag = (Node.MAX_MAG - current_mag)/10;
        new_mag = current_mag - delta_mag;
        tn.set_mag_no_history(new_mag);
    }
    // ----- end for all nodes -----

    GUI.P(0,"sharpen", "Done.");
    return(max_mag);
} // end sharpen

/**
 ** find_common_nodes_to FORGET THIS
 **/
public void find_common_nodes_to (Node node) {
    int i, size;
    Node tn;

    size = node.links.size();
    for (i=0; i<size; i++) {
        tn = (Node)(node_vec.elementAt(i));
        if (tn.isAN()) {
            GUI.P(0,"find_common_nodes_to","Got AN named "+tn.Name);
        }
    }
    // end find_common_nodes_to

    /**
    ** DataSea.connect_up
    **/
    public void connect_up (Node node) {
        Node tn;
        GUI.P(0,"connect_up", "Connecting up node "+node.Name);
    } // end connect_up

    /**
    ** DataSea.add
    **/
    public int add_to_unprocessed_vec (Node dn) {
        if (unprocessed_vec == null)
            unprocessed_vec = new Vector(100,10);
        unprocessed_vec.addElement(dn);
        GUI.P(0,"DS.add_to_unprocessed_vec","added node "+dn.Name+" size="+
            unprocessed_vec.size());
    }
}
```

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin

### DataSea.java

16

```
        return(0);
    } // end add_to_unprocessed_vec

/**
 ** DataSea.PostProcessor
 */
public void PostProcessor () {
    // add code to have this read a vector of nodes which haven't been processed yet,
    // from vector unprocessed_vec, probably added to by 'DataSea.words_input'
    Node tn;

    tn = (Node)(unprocessed_vec.elementAt(0));
    connect_up(tn);
    unprocessed_vec.removeElementAt(0);
} // end PostProcessor

/**
 ** DataSea.add_POV
 ** Create a new POV
 */
public void add_POV () {
    Node tn;

    // add code to have this read a vector of nodes which haven't been processed yet,
    // from vector unprocessed_vec, probably added to by 'DataSea.words_input'
    if (POV == null) {
        POV_namer++;
        POV = new Node("POV"+POV_namer, "POV",
            "", 0, 0, 20, 10);
        //POV.isPolarized = false;
        POV.z = 0;
        GUI.P(0,"DataSea.add_POV","POV created: "+POV.Name);
    }
    else {
        GUI.P(0,"DataSea.add_POV","POV already exists: "+POV.Name);
    }
} // end add_POV

/**
 ** DataSea.absorb_POV
 **
 */
public void absorb_POV (boolean save) {
    Node child = null;
    if (POV == null) {
```

```
        GUI.P(1,"absorb_POV","POV is null");
        return;
    }

    if (save) {
        GUI.P(1,"absorb_POV","Saving the POV as a Context Node");
        if (!POV.Name.substring(0,3).equals("POV")) {
            GUI.WARNING(0,"absorb_POV","POV.Name.substring(0,3) is "+POV.Name
                .substring(0,3));
            GUI.ERROR(0,"absorb_POV","POV.Name isn't POVxxx: "+POV.Name);
        }
        else {
            POV.Name = "P"+POV_namer;
            POV.isPOV = false;
            POV.setType("CN");
            GUI.P(1,"absorb_POV","Setting POV to null, Saved Name is <"+POV.Name+">, Saved
                Type is <"+POV.Type+">");
            child = POV.getNodeAtLink(0);
            if (child != null)
                GUI.P(1,"absorb_POV","POV.Link[0] is "+child.Name);
            else
                GUI.P(1,"absorb_POV","POV.Link[0] is NULL");
        }
    }
    POV.unlink_all();
    if (save) {
        POV.link(child);
        save_links(POV, POV);
    }
    POV = null;
    needdistUpdate=true;
    return;
} // end absorb_POV

/**
 ** save_links
 **
 */
public void save_links (Node node, Node CNnode) {
    int i, size;
    Node child;

    size = node.links.size();
    for (i=0; i<size; i++) {
        child = node.getNodAtLink(i);
```

```
        if (child.mag > 5) {
            save_links_r(node, child, CNode);
        }
    } // end save_links

    /**
     ** save_links_r
     **
     */
    public void save_links_r (Node node, Node child, Node CNode) {
        int i, size;
        Node grand_child=null;

        if (child.mag <= 5)
            return;

        GUI.P(1, "save_links_r", "Parent="+node.Name+", Child="+child.Name+", CNode="+CNode.Name
        e);
        addCNodeBetweenNodes(node, child, CNode);
        size = child.Links.size();
        for (i=0; i<size; i++) {
            grand_child = child.getNodeAtLink(i);
            if ((grand_child.mag > 5) && (grand_child.dist == 1+child.dist))
            {
                save_links_r(child, grand_child, CNode);
            }
        } // end save_links_r
    }

    /**
     ** DataSea.more_attraction
     **
     */
    public double more_attraction (boolean inverse_result) {
        if (inverse_result == true)
            gui.magscale /= 1.5;
        else
            gui.magscale *= 1.5;
        return(gui.magscale);
    }

    /**
     ** DataSea.more_repulsion
     **
     */
    /**
     **
     */
    public double more_repulsion (boolean inverse_result) {
        if (inverse_result == true)
            gui.TheiaMultiplier -= .1;
        else
            gui.TheiaMultiplier += .1;
        return(gui.TheiaMultiplier);
    }

    /**
     ** set_should_we_pos
     **
     */
    public void set_should_we_pos () {
        int size, isize, i, j;
        Node tnode, child;

        size = node.vec.size();
        for (i=0; i<size; i++) {
            tnode = (Node)node.vec.elementAt(i);
            isize = tnode.Links.size();
            for (j=0; j<isize; j++) {
                child = tnode.getNodeAtLink(j);
                tnode.set_links_should_we_pos(child, true); // set link tnode -> child true
            }
        } // end set_should_we_pos

    }

    /**
     **
     **
     **/
    void clear_Tdist ()
    Node tnode;

    int size, i;
    GUI.P(1, "clear_Tdist", "Clearing node_vec Tdists.");
    size = node.vec.size();
    for (i=0; i<size; i++) {
        tnode = (Node)node.vec.elementAt(i);
```

05/23/00  
00:48:20

DataSea.java

18

```

        inNode.Tdist = -1;
    }
    return;
} // end clear_Tdist

/*
**
**
**/
void clear_dist ()
Node inNode;

int size, i;
GUI.P(1, "clear_dist", "Cleaning node_vec dists.");
size = node_vec.size();
for (i=0; i<size; i++) {
    inNode = (Node)node_vec.elementAt(i);
    inNode.set_dist(-1);
}
return;
} // end clear_dist

/**
** calc_dist_between
**
**/
public double calc_dist_between (Node node1, Node node2) {
    int i, size;
    Node in;

    set_Tdist_start(node1);
    GUI.P(0, "calc_dist_between", "+node2.Name+" is "+node2.Tdist+" from "+node1.Name);
    return(node2.Tdist);

} // end calc_dist_between

/**
** newestest
**
**/
public void newestest () {
    int i, size;
    Node child;

    Node ruth = find_node_named("Ruth");

    show(ruth);
}

```

---

```

Node A1 = find_node_named("A1");
Node A2 = find_node_named("A2");
Node A3 = find_node_named("A3");
Node A4 = find_node_named("A4");
Node B1 = find_node_named("B1");
Node target = find_node_named("target");
if (POV==null) {
    GUI.WARNING(0, "DataSea.newest", "Need a POV");
    return;
}

set_Tdist_start(target);
System.out.println("A1.dist/Tdist=(" + A1.dist + ", " + A1.Tdist + ")");
System.out.println("A2.dist/Tdist=(" + A2.dist + ", " + A2.Tdist + ")");
System.out.println("A3.dist/Tdist=(" + A3.dist + ", " + A3.Tdist + ")");
System.out.println("B1.dist/Tdist=(" + B1.dist + ", " + B1.Tdist + ")");
System.out.println("A4.dist/Tdist=(" + A4.dist + ", " + A4.Tdist + ")");
System.out.println("target.dist/Tdist=(" + target.dist + ", " + target.Tdist + ")");

} // end newestest

/**
** check_for_loop_beginning
**
**/
public boolean check_for_loop_beginning (Node caller, Node node) {
    int i, size;
    Node child=null;
    boolean ret=false;

    size = node.links.size();
    for (i=0; i<size; i++) {
        child = node.getNodeAtLink(i);
        if (child != caller) {
            if (child.dist <= 1) // We've gone all the way back to the POV
                return(ret);
            if (child.dist < node.dist) {
                System.out.println("CFLB: (child<node) N("+describe(node)+")---->C("+describe(node
                ist(child)+")");
                if (true==check_for_loop_beginning(node, child)) {
                    ret = true;
                }
            }
            node.Tdist = child.Tdist + 1; // if branch has high value of Tdist, propagate it
        }
        else
            if (child.dist > node.dist) { // May be on branch with reversal inside it

```

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin

DataSea.java

19

```
System.out.println("CFLB: (child>node) N("+describe_node_dist(node)+"---sense_rev-->C("+de
scribe_node_dist(child)+"");
    if (true==sense_rev(node, child)) { // sense_rev will back up and increase Tdist
        *
        ret = true;
        node.Tdist = child.Tdist + 1; //continue increasing Tdist's from sense
    ev
    System.out.println("CFLB: (child>node) N("+describe_node_dist(node)+"---increasing Tdist after
sense_rev-->C("+describe_node_dist(child)+"");
    }
    }
    return(ret);
    } // end check_for_loop_beginning

/**
 * sense_rev
 */
public boolean sense_rev (Node caller, Node node) {
    int i, size;
    Node child;
    boolean ret = false;

    size = node.links.size();
    for (i=0; i<size; i++) {
        child = node.getNodeAtLink(i);
        if (child != caller) {
            if (child.dist > node.dist) {
                System.out.println("sense_rev: (child>node) N("+describe_node_dist(node)+"--->C("+describe_n
ode_dist(child)+"");
                ret = sense_rev(node, child);
                if (ret) { // IF TRUE THEN WE ARE ON THE PATH THAT NEEDS TO B
                    E CHANGED
                    return(ret);
                }
            }
            if (child.dist <= node.dist) { // FOUND REVERSAL POINT, INFORM CALLER THA
                T WE DID
                System.out.println("sense_rev: (child<=node) N("+describe_node_dist(node)+" REVERSAL C("+
describe_node_dist(child)+"");
                return(true); // don't change this yet
            }
        }
    }

    return(ret);
    } // end sense_rev
}

// HERE HERE
/**
 * describe_node_dist
 */
public String describe_node_dist (Node node) {
    int i, size;
    Node child;
    return(node.Name+"("+node.dist+" "+node.Tdist+"");
    } // end describe_node_dist

/**
 * set_Tdist_start One-node version
 */
void set_Tdist_start (Node parent) {
    int level=0;
    boolean complete=false, ret;

    if (parent==null) {
        GUI.WARNING(0, "set_Tdist_start()", "parent==null");
        return;
    }
    GUI.P(0, "set_Tdist_start()", "Starting on "+parent.Name);
    clear_Tdist();
    parent.Tdist = 1; // force this first one, then use recursion for others
    level=2;
    while (complete == false) {
        complete = true;
        gui.p("Tdist_start: level="+level);
        ret = set_Tdist_recursive(parent, level);
        if (ret == false)
            complete = false;
        level++;
    }
    return;
    } // end set_Tdist_start
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin DataSea.java

20

```
/*
** set_Tdist_recursive
**
** This increments a node's .Tdist whatever type it is
** Return true only when we've run out of children
** or when none of the children are at 1+caller.Tdist
** complete starts out true, it conditionally only changes to fa
lsc.
**/
boolean set_Tdist_recursive (Node node, int level) { // position this node's children
int i, size, Tdist;
boolean ret=true, complete=true;
Node child;
// search for where we were last, where Tdis were set to (level-1)

if (node.Tdist == level) {
    complete=false;
    return(complete);
}

size = node.Links.size();
for (i=0; i<size; i++) { // check Tdist of all children, position if appropriate
    child = node.getNodeAtLink(i);
    if (child == Root) { // Don't go back through the POV
        return(complete);
    }
    if (child == POV) { // Don't go back through the POV
        return(complete);
    }
    if (child.Tdist == -1) {
        child.Tdist = ((int)(node.Tdist)+1);
        complete = false;
    }
    else if (child.Tdist > node.Tdist) { // RECURSE
        ret=set_Tdist_recursive(child, level);
    }
    if (ret==false)
        complete=false;
}

return(complete);
} // end set_Tdist_recursive
/*
```

```
*/
void set_dist_start (Node parent) {
    int level=0;
    boolean complete=false, ret;

    if (parent == null) {
        GUI.WARNING(0,"set_dist_start(1)", "parent is null");
        return;
    }
    GUI.P(0,"set_dist_start(1)","Starting on "+parent.Name);
    clear_dist();
    parent.set_dist(1); // force this first c.re, then use recursion for others
    parent.set_Tdist(1); // force this first one, then use recursion for others
    level=2;
    while (complete == false) {
        GUI.P(0,"set_dist_start","iterating on level="+level);
        complete = true;
        // ret = set_dist_recursive(parent, level);
        ret = new_set_dist_recursive(parent, 1, level, 0);
        if (ret == false)
            complete = false;
        level++;
    }
    return;
} // end set_dist_start

/*
**
** // NEW VERSION
boolean new_set_dist_recursive (Node parent, int current_level, int target_level, int AN_level) { // po
sition this parent's children
int i, size, new_AN_level=AN_level;
boolean ret=true, complete=true, same_type=false, recurse=false;
Node child;
double delta=0.1, new_dist=-1;

// search for where we were last

if (current_level >= target_level)
{
    complete=false;
    return(complete);
}

// HERE
if (POV != null)
```

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin

DataSea.java

21

```
if (parent.StopsSpread && parent.dist != (POV.dist+1)) // Don't go back through stopped nodes
    return(complete);
```

```
if (parent.isAN)
    parent.AN_level = AN_level;
```

```
// HERE
size = parent.Links.size();
for (i=0; i<size; i++)
```

```
{
    child = parent.getNodeAtLink(i);
```

```
    ret = true;                // set defaults for next loop
    same.Type = false;         // set defaults for next loop
    recurse = false;           // set defaults for next loop
```

```
if (parent.Type == child.Type)
    same.Type = true;
```

```
if (same.Type)
    new_dist = parent.dist + delta;
else
```

```
    new_dist = (int)(parent.dist) + 1; // add one to truncated value
```

```
if (child.dist == -1) { // just beginning
    child.set_dist(new_dist);
    child.set_Tdist(parent.Tdist + 1);
    complete=false;
    recurse=false; // Done, don't recurse. This slowed me down.
}
```

```
else // if already the optimal distance, continue on this path until level is met
if (((child.dist < new_dist + 0.000001) && (child.dist > new_dist - 0.000001))
```

```
{
    recurse=true;
}
```

```
if (child.isAN)
    new_AN_level = AN_level+1;
```

```
if (recurse)
    ret=new_set_dist_recursive(child, current_level+1, target_level, new_AN_level);
```

```
if (ret==false) // one instance of ret==false makes the whole thing false, we keep on
    complete=false;
```

```
return(complete);
```

```
} // end NEW VERSION set_dist_recursive
```

```
/*
**
```

```
** // MIDDLE VERSION
```

```
boolean set_dist_recursive (Node parent, int level) { // position this parent's children
    int i, size, dist;
```

```
    boolean ret=true, complete=true, recurse=false;
    Node child;
    double new_dist=-1;
```

```
// search for where we were last, where dists were set to (level-1)
```

```
if (parent.dist == level)
```

```
{
    complete=false;
    return(complete);
}
```

```
// HERE
```

```
if (POV != null)
```

```
if (parent.StopsSpread && parent.dist != (POV.dist+1)) // Don't go back through stopped nodes
    return(complete);
```

```
size = parent.Links.size();
for (i=0; i<size; i++)
```

```
{ // check dist of all children, position if appropriate
    child = parent.getNodeAtLink(i);
```

```
if (child.dist == -1) {
```

```
    //if (parent.Type == child.Type)
    if (parent.isDN && child.isDN)
        {new_dist = parent.dist + 0.1;}
```

```
    else
        {new_dist = parent.dist + 1; }
    parent.set_links_VRparamsTo(child);
    complete = false;
```

```
}
else {
```

```
if (parent.isDN && child.isDN) {
    //if (parent.Type == child.Type) { }
```

05/23/00  
00:48:20

Confidential and Proprietary Property of Rocky Nevin  
DataSea.java

22

```
        if (child.dist >= (parent.dist+0.1)) {
            {new_dist = parent.dist + 0.1; recurse = true;}
        }
    }
    //System.out.println(parent.Name+" "+child.Name+" p_dist="+parent.dist);

    else // not same type
    {
        if (((int)(child.dist) >= ((int)parent.dist+1))) {
            {new_dist = parent.dist + 1; recurse = true;}
        }
    }

    }

    if (child.Name.equalsIgnoreCase("mtg") || child.Name.equalsIgnoreCase("12 Mar 2000")) {
        System.out.println(parent.Name+" "+child.Name+" p_dist="+parent.dist
            +", old_child_dist="+child.dist
            +", new_child_dist="+new_dist+" recurse="+recurse);
    }
    child.set_dist(new_dist);
    if (recurse)
        ret=set_dist_recursive(child, level);
    recurse = false; // reset it
    if (ret==false)
        complete=false;
    }
    return(complete);
} // end set_dist_recursive
**/

/*
**
**// OLD VERSION
boolean set_dist_recursive (Node parent, int level) { // position this parent's children
    int i, size, dist;
    boolean ret=true, complete=true;
    Node child;
    // search for where we were last, where dists were set to (level-1)

    if (parent.dist == level)
    {
        complete=false;
        return(complete);
    }

    // HERE
    if (POV != null)
    if (parent.StoodSoread && parent.dist != (POV.dist+1)) // Don't go back through stopped nodes

        if (child.dist >= (parent.dist+0.1)) {
            return(complete);
        }
        size = parent.links.size();
        for (i=0; i<size; i++)
        { // check dist of all children, position if appropriate
            child = parent.getNodeAtLink(i);

            if (child.dist == -1) {
                child.set_dist((parent.dist+1));
                parent.setLinksVRparamsTo(child);
                complete = false;
            }
            else if (child.dist == (parent.dist+1))
                ret=set_dist_recursive(child, level);
            if (ret==false)
                complete=false;
        }
    }
    return(complete);
} // end set_dist_recursive

**/
** OKioFollow returns boolean whether to invoke action on grand_child,
** looks at three nodes for context
*/
public boolean OKioFollow (Node parent, Node child, Node grand_child) {
    boolean ret, boolean = false;

    if (parent == POV) // sometimes POV is too small for test below of mag
        return(true);
    if (parent == null)
        return(true);
    if (child == null)
        return(true);
    if (grand_child == null)
        return(true);

    // ((child.isDN && !grand_child.isDN) || gui.drawDN) &&
    // ((child.isAN && !grand_child.isAN) || gui.drawAN) &&
    // ((child.isCN && !grand_child.isCN) || gui.drawCN) &&
    // ((child.isPN && !grand_child.isPN) || gui.drawPN) &&
    // ((gui.checkPolarization && parent.isBB && !child.isBB)) &&
    // ((child.isEvent && !grand_child.isEvent) || gui.drawEvent)

    if (child.mag >= 0.2)
        ret, boolean = true;
}
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

DataSea.java

23

```
return(ret, boolean);

} // end OKtoFollow

/*
** set_ChildNum_start Set the ChildNum variable, by calling set_ChildNum_recursive
**/
void set_ChildNum_start (Node parent, Node node, boolean theta_org_flag) {
    int i, vec1=0;

    if (node == null)
        return;

    GUI.P(1, "set_ChildNum_start", "Starting on "+node.Name);
    set_ChildNum_recursive(parent, node, theta_org_flag);
    return;
} // end set_ChildNum_start

/*
** set_ChildNum_recursive We assume the dist has been properly set, now we
** just count the number of children having dist = node.dist+1
** Returns max number of children to the caller, node
**/
void set_ChildNum_recursive (Node parent, Node node, boolean theta_org_flag) { // set the children
    's_ChildNum
    int i, j, size, ChildNum=0, iChildNum=0, ChildCount;
    Node child, tn;
    Node node1, node2;

    int BigChildCount = 0;

    String parentName = "unassigned";

    if (parent != null)
        parentName = parent.Name;

    if (gui.Debug==2) {
        if (parent==null)
            System.out.println("SCN: set_ChildNum() START: "+node.Name);
        else
            System.out.println("SCN: "+parent.Name+"->"+node.Name);
    }

}

//
// Calculate BigChildCount based on all siblings (rather than distal children)
size = node.Links.size();
for (i=0; i<size; i++)
{
    child = node.getNodeAtLink(i);
    if (child.mag > Node.BIG_MAG)
        BigChildCount++;
}
node.BigChildCount = BigChildCount;

for (i=0; i<size; i++)
{
    child = node.getNodeAtLink(i);
    if (((child.dist > node.dist) && child!=Root) // This child'll be drawn next
        //
        {
            if (ChildNum < Node.MAX_CHILDREN)
                //
                if (OKtoFollow(parent, node, child))
                {
                    node.child_vec[ChildNum] = i; // set starting with 0
                    child.ChildNum = ++ChildNum; // starts with 1
                    set_ChildNum_recursive(node, child, theta_org_flag);
                }
            }
        }
    }
    node.ChildCount = ChildNum;
    ChildCount = node.ChildCount;

    if (ChildCount >= Node.MAX_CHILDREN) {
        GUI.WARNING(0, "set_ChildNum_recursive", "ChildCount >= Node.MAX_CHILDREN
        , bailing.");
        return;
    }

}

/*****
if (theta_org_flag) {
// Now, order the vector child_vec
for (j=0; j<ChildCount-2; j++)
for (i=0; i<ChildCount-1-j; i++) {

// SWAP AND LATER RESET CHILDNUM

```

```
node1 = node.getNodeAtLink( node.child_vec[i] );
node2 = node.getNodeAtLink( node.child_vec[i+1] );

if (node1.mag < node2.mag) {
    x = node.child_vec[i];
    node.child_vec[i] = node.child_vec[i+1];
    node.child_vec[i+1] = x;
    x = node1.ChildNum;
    node1.ChildNum = node2.ChildNum;
    node2.ChildNum = x;
    GUI.P(1,"set_ChildNum_recursive",j,"j+")swapping nodes "+node.child_vec[i].Name+
    "<->" +node.child_vec[i+1].Name);
}
} // end theta_org_flag

for (i=0; i<ChildCount; i++) {
    node1 = node.getNodeAtLink( node.child_vec[i] );
    node1.ChildNum = i+1; // ChildNum starts with 1, the index with 0
    /*****
    public void calc_mags (Node target,node) {
        double mag1=0, mag2=0, avg_mag=0;
        Node tn, node1, node2;
        int size, i;

        size = target.node.Links.size();
        switch (size) {
            case 0: break; // ought not to happen
            case 1: avg_mag = 0.5*(target.node.getNodeAtLink(0)).mag;
                    if (avg_mag > target_node.mag)
                        target_node.set_mag(avg_mag);
                    break;
            case 2: mag1 = (target_node.getNodeAtLink(0)).mag;
                    mag2 = (target_node.getNodeAtLink(1)).mag;
                    target_node.set_mag(0.5*(mag1 + mag2));
                    break;
        }

        default:
            for (i=0; i<size; i++) { // find avg of two largest mag's
                tn = target_node.getNodeAtLink(i);
                if (tn.mag > mag1)
                    mag1 = tn.mag;
                else if (tn.mag > mag2)
                    mag2 = tn.mag;
            }
            avg_mag = 0.5*(mag1 + mag2);
            if (avg_mag > target_node.mag)
                target_node.set_mag(avg_mag);
            break;
        }

        GUI.P(1,"calc_mags", "node "+target_node.Name+" mag="+target_node.mag);
    } // calc_mags

    /**
    ** backup
    **
    */
    public void backup () {
        int i, size;
        Node parent, tparent;

        if (gui.lastNode == null)
            return;

        parent = gui.lastNode;
        do {
            tparent = parent.getParent();
            if (tparent==null)
                break;
            if (tparent.dist > 1)
                parent = tparent;
            // don't go back to the POV,
            // but do allow going back one beyond the same
            type
            if (tparent.Type != gui.lastNode.Type)
                break;
            } while (true);

        // now, the parent is prior to the last one of the same type as the starting point.
        gui.lastNode = parent;
    }
```

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin

DataSea.java

25

```
} // end backup

/*
** potmag potentiate distal to target, mag distal to POV
** spread initially both polarizations, halt at pol-transition
** touch !ANs but don't recurse.
** Then, mag distally from another high-order AN.
*/
public void potmag (String words[], int num_words) {
    int size, i;
    Node pot_target=null, mag_target=null;

    if (num_words == 1)
        pot_target = GUI.lastNode;
    else
        if (num_words == 2) {
            pot_target = find_node_named(words[1]);
            mag_target = find_node_named(words[2]);
        }
    else
        if (num_words == 3) {
            pot_target = find_node_named(words[1]);
            mag_target = find_node_named(words[2]);
        }

    if (pot_target == null) {
        GUI.WARNING(0, "potmag", "Need a lastNode");
        return;
    }

    if (pot_target == null) {
        GUI.WARNING(0, "potmag", "Need a name or at least a lastNode. POV sibling is implied.");
        return;
    }

    if (POV != null) {
        if (mag_target == null) // if no second arg is given, assume its the first relative of POV
            mag_target = POV.getNodeAtLink(0);
        if (mag_target == null) {
            GUI.WARNING(0, "potmag", "Need a mag_target");
            return;
        }
    }

    System.err.println("----- Beginning of potmag() -----");
    GUI.P(0, "potmag", "Using pot_target "+pot_target.Name+", mag_target of "+mag_target.Name);
    potmag(pot_target, mag_target);
    System.err.println("----- End of potmag() -----");
    return;
} // end potmag

public void potmag (Node pot_target, Node mag_target) {
    // SET THE POTENTIATION DISTAL FROM TARGET NODE (using Tdist)
    /*****
    ** set_Tdist.start(pot_target); // NEED FOR SETTING VARIABLE "TDIST"
    ** pot(pot_target, true); // USES "TDIST"
    *****/
    pot(pot_target); // sets Tdist from Thesaurus_node
    // SET THE MAG DISTAL FROM PRIMARY SIBLING OF POV
    magall(mag_target); // USES "DIST"

    return;
} // end potmag

public void poth (Node node) { //
    int size, i;
    double depth, up_depth, down_depth;
    Node tn=null;

    if (node == null)
        return;

    if (Thesaurus_node == null) {
        GUI.WARNING(0, "poth", "Need Thesaurus_node. Aborted.");
        return;
    }

    set_Tdist.start(Thesaurus_node); // NEED FOR SETTING VARIABLE "TDIST"

    // Check current depth of node
    depth = node.Tdist;
    up_depth = depth / 2;
    down_depth = 100; // go all the way down always

    gui.p("poth"+gui_node.to_string(node)+"");
    node.set_pot(GUI.current_TS);

    // SET THE POTENTIATION DISTAL FROM SECONDARY NODE
```

```
size = node.links.size();
for (i=0; i<size; i++) {
    tn = node.getNodeAtLink(i);
    // SEE IF CHILD IS DOWNHILL
    if ((tn.Tdist == 1+node.Tdist)) { // going downhill ...
        poth_r(tn, "down", down_depth);
    }
    else
        // OR UPHILL
        if ((tn.Tdist == -1+node.Tdist)) { // going uphill ...
            poth_r(tn, "up", up_depth);
        }
    }
    return;
} // end poth

/**
 ** poth_r
 **
 */
public void poth_r (Node node, String direction, double target_depth) {
    int i, size, delta;
    Node child;

    if (node.Debug)
        GUI.P(0, "poth_r", "setting pot on "+node.Name + ", Tdist="+node.Tdist+", Type="+node.Type);
    node.set_pot(GUI.current_TS);

    // Bail out if we are at the target depth
    if (node.Tdist == target_depth) {
        return;
    }

    if (direction.equalsIgnoreCase("up"))
        delta = -1;
    else
        if (direction.equalsIgnoreCase("down"))
            delta = 1;
    else {
        GUI.WARNING(0, "poth_r", "Wrong argument for direction passed, is "+direction);
        return;
    }
}

size = node.links.size();
for (i=0; i<size; i++) {
    child = node.getNodeAtLink(i);
    if (node.Tdist + delta == child.Tdist) // recurse if in the correct direction
        poth_r(child, direction, target_depth);
    }
} // end poth_r

/**
 ** lower_all_pots
 **
 */
public void lower_all_pots (Node node) {
    int i, size;
    Node child;

    size = node.vec.size();
    for (i=0; i<size; i++) {
        child = (Node)(node.vec.elementAt(i));
        child.set_pot(GUI.current_TS);
    }
} // end lower_all_pots

/**
 */
// method pot 2nd version distributes pot_r considering polarization of start
//
public void pot (String words[], int num_words) {
    Node node=null;

    if (num_words == 1)
        node = gui.lastNode;
    else
        if (num_words > 1)
            {
                node = find_node_named(words[1]);
            }
        if (node == null) {
            GUI.WARNING(0, "pot", "Need a name or at least a lastNode");
            return;
        }
    GUI.P(1, "pot", "Top-level. Running on "+node.Name);
}
```

05/23/00  
00:48:20

Confidential and Proprietary Property of Rocky Nevin  
DataSea.java

27

```
//lower_all_pots();
pot(node, false); // if given as command, don't recurse
return;
} // end pot

public void pot (Node node, boolean recurse) { // distributes pot_r considering polarization of star
{
    int size, i;
    Node tn=null;

    if (node == null)
        return;

    // SET THE POTENTIATION DISTAL FROM SECONDARY NODE
    size = node.links.size();
    for (i=0; i<size; i++) {
        tn = node.getNodeAtLink(i);
        tn.set_pot((GUI.current_TS);
        GUI.P(0,"pot","on "+tn.Name
        +", Tdist="+tn.Tdist+", Type="+tn.Type);
        if (recurse && (tn.Tdist > node.Tdist) { // recurse only if child.Tdist > us
            pot_r(tn, node.getPol(tn), tn.Type); // pass on the polarization
        }
    }
    return;
} // end pot

public void pot_r (Node node, char original_pol, String original_type) {
    int size, i;
    Node tn=null;

    if (node == null)
        return;

    // SET THE POTENTIATION DISTAL FROM SECONDARY NODE
    size = node.links.size();
    for (i=0; i<size; i++) {
        tn = node.getNodeAtLink(i);
        tn.set_pot((GUI.current_TS);
        if ((tn.Tdist == 1+node.Tdist)
            && (tn.Type.equals(original_type))
            && (original_pol==node.getPol(tn)))
            { // recurse
                GUI.P(1,"pot_r",node.getPol(tn)+" Type="+tn.Type
                +", Tdist="+tn.Tdist+" "+node.Name+" calling "+tn.Name);
                pot_r(tn, original_pol, original_type);
            }
            else { // not recursing
                //GUI.P(1,"pot_r","NOT RECURSING "+node.getPol(tn)+" Type="+tn.Type
                // +", Tdist="+tn.Tdist+" "+node.Name+" calling "+tn.Name);
            }
        }
    }
    return;
} // end pot_r

/**
 ** method back_p
 */
public void back_p (String words[], int num_words) {
    Node node=null;

    if (num_words==0)
        return;

    if (num_words==1)
        node = gui.lastNode;
    else
        node = find_node_named(words[1]);

    if (node == null)
        return;

    GUI.P(0,"back_p", "Operating on "+node.Name);
    back_p(node); // recursive fn
} // end back_p (String, int)

public void back_p (Node node) {
    int size, i;
    Node tn=null;

    size = node.links.size();
    for (i=0; i<size; i++) {
        tn = node.getNodeAtLink(i);
        if (tn.dlst < node.dlst) {
            GUI.P(0,"back_p", "i is "+i+", tn="+tn.Name);
        }
    }
}
```

05/23/00  
00:48:20

Confidential and Proprietary Property of Rocky Nevin  
DataSea.java

28

```
in.set_pos(GUI.current_TS);
// tn.potentialion_TS = GUI.current_TS; // p_mag determined from this
back.p(tn); // recurse
}
} // end back.p

/**
** method upurls magnify the URLs upstream from each URL
*/
public void upurls (String words[], int num_words) {
    int target_level=0, size, i, child_size, child_i;
    Node node=null, child;
    Link link=null;
    boolean only = false; // do the target level and distal ones

    gui.P(0,"upurls","Begun.");

    size = node.vec.size();
    for (i=0; i<size; i++) {
        node = (Node)(node.vec.elementAt(i));
        if (node.isURL) {
            // ----- for children -----
            child_size = node.links.size();
            for (child_j=0; child_j<child_size; child_j++) {
                child = (Node)(node.getNodeAtLink(child_j));
                if (child.isURL) {
                    link = node.getLinkTo(child);
                    if (child == link.Node) {
                        gui.P(0,"upurls","child.Name+" magnified by "+
                            //child.more_mag(); // FINALLY, MAG THE H
                            child.set_mag(child_mag * node_mag); // FINAL
                    }
                }
            }
            // ----- end for children -----
        }
    }

    gui.P(0,"upurls","Done.");
    return;
} // end upurls

/**
** set_selected_on_big_nodes
**
*/
public void set_selected_on_big_nodes () {
    int i, size;
    Node child;

    size = node.vec.size();
    for (i=0; i<size; i++) {
        child = (Node)(node.vec.elementAt(i));
        if (child.isAN && (child_TS > GUI.current_TS-10000))
            child.isSelected = true;
    }
    return;
} // end selectrecent

/**
** FlattenANs set all ANs to MED_MAG
**
*/
public void FlattenANs () {
    int i, size;
    Node child;

    size = node.vec.size();
    for (i=0; i<size; i++) {
        child = (Node)(node.vec.elementAt(i));
        if (child.isAN && (child_TS > GUI.current_TS-10000))
            child.isSelected = true;
    }
    return;
} // end selectrecent

/**
** FlattenANs set all ANs to MED_MAG
**
*/
public void FlattenANs () {
    int i, size;
    Node child;

    size = node.vec.size();
    for (i=0; i<size; i++) {
        child = (Node)(node.vec.elementAt(i));
        if (child.isAN && (child_TS > GUI.current_TS-10000))
            child.isSelected = true;
    }
    return;
} // end selectrecent

/**
** FlattenANs set all ANs to MED_MAG
**
*/
public void FlattenANs () {
    int i, size;
    Node child;

    size = node.vec.size();
    for (i=0; i<size; i++) {
        child = (Node)(node.vec.elementAt(i));
        if (child.isAN && (child_TS > GUI.current_TS-10000))
            child.isSelected = true;
    }
    return;
} // end selectrecent

/**
** FlattenANs set all ANs to MED_MAG
**
*/
public void FlattenANs () {
    int i, size;
    Node child;

    size = node.vec.size();
    for (i=0; i<size; i++) {
        child = (Node)(node.vec.elementAt(i));
        if (child.isAN && (child_TS > GUI.current_TS-10000))
            child.isSelected = true;
    }
    return;
} // end selectrecent
```

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin

DataSea.java

29

```
gui.P(0,"flattenANs","Begun.");

size = node.vec.size();
for (i=0; i<size; i++) {
    child = (Node)(node.vec.elementAt(i));
    if (child.isAN())
        child.set_mag(Node.MED_MAG);
}

gui.P(0,"flattenANs","Done.");

return;
} // end flattenANs

/**
 ** ans show the ANs two links from lastNode, force their mag to MAX_MAG+0.1
 **
 */
public void ans () {
    int i,j, size_i, size_j;
    Node child, grand_child;

    Node node = gui.lastNode;

    size_i = node.Links.size();
    for (i=0; i<size_i; i++) {
        child = node.getNodeAtLink(i);
        size_j = child.Links.size();
        for (j=0; j<size_j; j++) {
            grand_child = child.getNodeAtLink(j);
            if (grand_child.isAN())
                grand_child.set_mag(Node.MAX_MAG+0.1);
        }
    }

    } // end ans

/**
 ** inhstored
 **
 */

public void inhstored () {
    int i, size;
    Node node;

    gui.P(0,"inhstored","Begun.");

    if (big_mag_node.vec == null) {
        gui.WARNING(0,"inhstored","big_mag_node.vec is null.");
        return;
    }

    size = big_mag_node.vec.size();
    for (i=0; i<size; i++) {
        node = (Node)(big_mag_node.vec.elementAt(i));
        node.set_mag(Node.SMALL_MAG);
    }

    gui.P(0,"inhstored","Lowered "+big_mag_node.vec.size()+" elements in big_mag_node.vec.");
    return;
} // end inhstored

/**
 ** storebig
 **
 */
public void storebig () {
    int i, size;
    Node child;

    big_mag_node.vec = null; // how do we release an object? 'free()' doesn't seem to work
    big_mag_node_vec = new Vector(10);

    size = node.vec.size();
    for (i=0; i<size; i++) {
        child = (Node)(node.vec.elementAt(i));
        if (child.isAN() && child.mag > Node.BIG_MAG)
            big_mag_node_vec.addElement(child);
    }

    gui.P(0,"storebig","There are "+big_mag_node_vec.size()+" elements in big_mag_node_vec.");
    return;
} // end storebig

/**
 ** method absURLs
 **
 */
```

```
public void absURLs () { // Assumes user has run selectrecent() and FlattenANsO.
    // Mag unselected AN's of URLs based on the URL's mag
    // Assume that large-mag AN's have been selected (based
    // perhaps on their being touched recently)

    int size, i;
    Node node;

    GUI.P(0, "absURLs", "Begun.");

    // ----- for all nodes -----
    size = node_vec.size();
    for (i=0; i<size; i++) {
        node = (Node)(node_vec.elementAt(i));
        if (node.isURL)
            if (node.mag > Node.BIG_MAG) {
                // set Tdist.start(node); // NEED FOR SETTING VARIABLE "TDIS
                T
                GUI.P(0, "absURLs", "Starting on URL <"+node.Name+"> mag="+
                node.mag);
                absURLs.r((Node)null, node);
                //prime_or_mag(node);
            }
        }
    } // ----- end for all nodes -----

    //needdistUpdate = true;
    return;
} // end absURLs

/**
** prime_or_mag Either prime this node (make it sensitive to another call) or mag it
** Called usually by fn meant to characterize DNS
*/
public void prime_or_mag (Node node) {
    int i, size;
    Node child;

    GUI.P(0, "prime_or_mag", "Called on "+node.Name+", dist="+node.dist);

    // Call from xabs() from URL, find AN children, invoke this fn on them

    size = node.Links.size();
    for (i=0; i<size; i++) {
        child = node.getNodeAtLink(i);

        if (node.isURL)
            if (child.isAN)
                if (child.potentialion_TS > (GUI.current_TS - 1000)) { // we've been touched recently
                    child.more_mag(node);
                    System.out.println("node.isURL & child.isAN & touched <"+node.Name+"> to <"+child
                    d.Name+">, dist="+child.dist);
                    prime_or_mag(child);
                }
            else {
                child.set_pot();
                System.out.println("node.isURL & child.isAN & NOT touched <"+node.Name+"> to <
                "+child.Name+">, dist="+child.dist);
            }
        }
        if (node.isAN)
            if (child.isAN)
                if (node.goesUpstreamTo(child))
                    if (child.potentialion_TS > (GUI.current_TS - 1000)) { // we've been touched recently
                        child.more_mag(node);
                        System.out.println("node.isAN & child.isAN & upstream & touched <"+node.Name+">
                        to <"+child.Name+">, dist="+child.dist);
                        prime_or_mag(child);
                    }
                else {
                    child.set_pot();
                    System.out.println("node.isAN & child.isAN & upstream & NOT touched <"+node.Na
                    me+"> to <"+child.Name+">, dist="+child.dist);
                }
            }
        }
    } // -----
    //if ((node.isAN && node.goesUpstreamTo(child)) || (node.isAN && child.dist > node.dist)) {
    //    if (child.potentialion_TS > (GUI.current_TS - 1000)) { // we've been touched recently
    //        if (child.isAN) {
    //            GUI.P(1, "prime_or_mag", "Mag'ing & recursing from "+node.Name+" to "+chi
    //            ld.Name+", dist="+child.dist);
    //            child.more_mag();
    //            prime_or_mag(child);
    //        }
    //        else
    //            GUI.P(1, "prime_or_mag", "NOT AN going from "+node.Name+" to "+child.Name+", dis
    //            t="+child.dist);
    //    }
}
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin DataSea.java

31

```
// else {
//     GUI.P(1,"prime_or_mag","BAD TS, pot'ing going from "+node.Name+" to "+
child.Name+", dist="+child.dist);
//     child.set_pot(0);
// }
// else
//     GUI.P(1,"prime_or_mag","REJECTING going from "+node.Name+" to "+chi
d.Name+", dist="+child.dist);
****: *****/

} // end prime_or_mag

/**
** method absURLs.r increment node.mag if node.Type=="AN", recurse only if child is AN
*/
public void absURLs.r (Node caller, Node node) {
    int size, i;
    Node tn;
    String caller_name = "<blank>";

// DON'T DO ANYTHING IF a caller is AN and child is not AN.
if (caller != null) {
    caller_name = caller.Name;
    if (caller.isAN && !node.isAN) {
        GUI.P(1,"absURLs.r","BAIL: (caller.isAN && !node.isAN) caller="+caller.
Name+", node="+node.Name);
        return;
    }
}

// DECIDE IF WE SHOULD MAG THIS
if (node.isAN) {
    GUI.P(0,"absURLs.r","MAGGING: "+node.Name
        +"",.substring(0,10)
        +"caller="+caller_name+"(dist="+caller.dist+" prime_or_mag()");
    caller.more_mag(node);
}

// DECIDE IF WE SHOULD RECURSE ON PARENT OF node
size = node.Links.size();
for (i=0; i<size; i++) {
    tn = node.getNodeAtLink(i);
    GUI.P(1,"absURLs.r","TESTING <"
        +node.Name+"->"
        +tn.Name);
    if (
        (node.isAN && tn.isAN) // if parent=AN, then to recurse, child must be AN
        || !node.isAN // if parent !AN, then child can be anything
    ) {
        if (tn.dist<node.dist) {
            GUI.P(1,"absURLs.r","RECURSING: from <"
                +node.Name+">("+node.dist+" to "
                +tn.Name+"("+tn.dist+"");
            absURLs.r(node, tn);
        }
        else
            GUI.P(1,"absURLs.r","NOT-RECURSING: from <"
                +node.Name+">("+node.dist+" to "
                +tn.Name+"("+tn.dist+"");
    }
}
return;
} // end absURLs.r

/**
** connect_all_to_POV
**
*/
public void connect_all_to_POV () {
    int i, size;
    Node child;

    if (POV==null)
        return;

    set_Tdist_start(POV); // NEED FOR SETTING VARIABLE "TDIST"
    size = node_vec.size();
    for (i=0; i<size; i++) {
        child = (Node)(node_vec.elementAt(i));
        if (child.mag > gui.relations.threshold) {
            gui.p(gui.node.to_string(child));
            threshold_back_r((Node)null, child);
        }
    }
}
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin DataSea.java

32

```
) // end connect_all_to_POV

/**
 ** method abs
 */
public void abs (String words[], int num_words) {
    int target_level=0;
    Node node;
    boolean only = false; // do the target_level and distal ones

    if (num_words==0)
        return;

    if (num_words==1)
        node = gui.lastNode;
    else
        node = find_node_named(words[1]);

    if (node == null) {
        gui.WARNING(0,"abs ", "node is null");
        return;
    }

    if (words[0].equalsIgnoreCase("abs"))
        target_level = 5;
    if (words[0].equalsIgnoreCase("abs1")) {
        target_level = 1;
        only = true;
    }
    if (words[0].equalsIgnoreCase("abs2")) {
        target_level = 2;
        only = true;
    }
    if (words[0].equalsIgnoreCase("abs3")) {
        target_level = 3;
        only = true;
    }
    if (words[0].equalsIgnoreCase("abs4")) {
        target_level = 4;
        only = true;
    }
    if (words[0].equalsIgnoreCase("abs5")) {

        target_level = 5;
        only = true;
    }
    if (words[0].equalsIgnoreCase("abs6")) {
        target_level = 6;
        only = true;
    }

    abs(node, target_level, only);
}

connect_all_to_POV();

// end abs

/**
 ** method abs
 */
public void abs (Node node, int target_level, boolean only) {
    int size, i;
    Node tn;
    boolean recurse=false;

    if (node == null)
        return;
    if (node == Root)
        return;
    node.lift(4); // Keep start high

    GUI.P(0,"abs ", "Operating on "+node.Name);

    // set Tdist_start(node); // NEED FOR SETTING VARIABLE "TDIST"
    size = node.Links.size();
    for (i=0; i<size; i++) {
        tn = node.getNodeAtLink(i);
        recurse = false; // reset
        if (((tn.dist==(int)node.dist+1))&&(tn.dist==node.dist+delta.dist)) {
            recurse = true;
            if (tn.isAN && node.isAN) { // if both AN, go upstream only
                if (node.goesUpstreamTo(tn))
                    recurse &= true;
            }
        }
    }
}
```

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin

DataSea.java

33

```
else
    recurse = false;
}
else
    recurse = false;
if (recurse) {
    abs.r(node, tn, target_level, 1, only); // start with this_level=0
}
GUI.P("abs", "NOT recursing from "+node.Name+" to "+tn.Name);
return;
} // end abs

/**
 * method abs.r
 * increment node.mag if node.Type=="AN", recurse only if child is AN
 */
public void abs_r (Node caller, Node node, int target_level, int t_is_level, boolean only) {
    int size, i;
    Node tn;
    boolean recurse=false;

    if (this_level > target_level)
        return;

    if (node.isAN) {
        if (only) { // only increase if exactly on target_level
            if (this_level == target_level)
                node.more_mag(caller);
        }
        else {
            node.more_mag(caller);
        }
        this_level++;
    }

    if (this_level > target_level)
        return;

    size = node.links.size();
    for (i=0; i<size; i++) {
        recurse = false; // reset
        tn = node.getNodeAtLink(i);
        if (
            (node.isAN && tn.isAN) // if parent=AN, then to recurse, child must be AN
            || !node.isAN // if parent !AN, then child can be anything
        ) {
            if (((tn.dist==(int)node.dist+1)
                || (tn.dist<=node.dist+delta_dist+0.00001
                    && tn.dist>=node.dist+delta_dist-0.00001)
                )) {
                recurse = true;
                if (tn.isAN && node.isAN) { // if both AN, go upstream only
                    if (node.goesUpstreamTo(tn)) {
                        recurse &= true;
                    }
                }
                else {
                    recurse = false;
                }
            }
            else {
                recurse = false;
            }
            if (recurse) {
                abs_r(node, tn, target_level, this_level, only);
            }
        }
        return;
    } // end abs_r

    /**
     * count_distal uses a 'global' variable, returns it: also used globally by functions
     */
    public int count_distal (Node node) {
        int return_value;

        GlobalDistalCount = 0;
        count_distal_r(node);

        return_value = GlobalDistalCount;
        GUI.P("count_distal", "Found "+return_value+" distal nodes.");
        return(return_value);
    } // end count_distal
}
```

```

/**
 ** count_distal_r
 **
 */
public void count_distal_r (Node node) {
    int i, size;
    Node child;

    if (node == null)
        return;

    size = node.links.size();
    for (i=0; i<size; i++) {
        GlobalDistalCount += size;
        child = (Node)(node.getNodeAtLink(i));
        if (child.dist > node.dist)
            count_distal_r(child);
    }
} // end count_distal_r

/**
 ** method sides chain data nodes by making those of same type in a chain visible
 */
public void sides (Node node) {
    int i, size;
    Node tn;

    if (node.isAN()) {
        size = node.links.size();
        for (i=0; i<size; i++) {
            tn = (Node)(node.getNodeAtLink(i));
            sides_r(tn, tn.Type);
        }
    }
    else
        sides_r(node, node.Type);
} // end sides

/**
 ** method sides_r chain data nodes by making them all visible
 */
public void sides_r (Node node, String type) {
    Node child=null;

    int i, size;
    if (node == null)
        return;
    if (node == Root)
        return;
    // RECURSE
    size = node.links.size();
    for (i=0; i<size; i++) {
        child = node.getNodeAtLink(i);
        // check for dis-similar type to magnify
        if ((child.dist > node.dist) && (child.Type.equalsIgnoreCase(type))) {
            pump(child);
            gui.getToolkit().sync(); // drama
            //gui.show_node_once(child);
        }
        // check for similar type to recurse on
        if ((child.dist > node.dist) && (child.Type.equalsIgnoreCase(type))) {
            System.err.println("sides_r:"+node.Name+"-> recursing on "+child.Name+" from "+node.Name);
            sides_r(child, type);
        }
    }
    return;
} // end sides_r

/**
 ** shiftIn
 **
 */
public void shiftIn (String[] words, int num_words) {
    Node CNode=null;
    if (null == (CNode = figure_out_node("shiftIn", words, num_words)))
        return;
    gui.P[0].shiftIn ("Running on CNode<"+CNode.Name+">");
    mag_CSs_of_CNode(CNode)//mag CS's and mag
} // end shiftIn

/**
 ** shiftOut

```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

## DataSea.java

35

```

**
*/
public void shiftOut (String[] words, int num_words) {
    Node CNode=null;

    if (null == (CNode = figure_out_node("shiftOut", words, num_words)))
        return;
    gui.P(0,"shiftOut","Running on CNode<"+CNode.Name+">");
    inh_CSs_of_CNode(CNode)/mag CS's and mag
    } // end shiftOut

/**
** shiftOutAll
**
*/
    public void shiftOutAll (String[] words, int num_words) {
        int size, j_size, i, j;
        Node node;
        Link siblink;

        size = node_vec.size();
        for (i=0; i<size; i++) {
            node = (Node)(node_vec.elementAt(i));
            j_size = node.Links.size();
            for (j=0; j<j_size; j++) {
                siblink = node.getLink(j);
                if (i<10)
                    GUI.P(0,"DataSea.shiftOutAll","resetting CS to "+Link.MIN_CS+
                        " for node <"+node.Name+"> and <"+siblink.NodeR.Nam
                        e+">");
                if (siblink!=null)
                    siblink.set_CS(Link.MIN_CS);
            }
        }
    } // end shiftOutAll

    /**
    ** shiftInAll
    **
    */
    public void shiftInAll (String[] words, int num_words) {
        int size, j_size, i, j;
        Node node;
        Link siblink;

        size = node_vec.size();
        for (i=0; i<size; i++) {
            node = (Node)(node_vec.elementAt(i));
            j_size = node.Links.size();
            for (j=0; j<j_size; j++) {
                siblink = node.getLink(j);
                if (i<10)
                    GUI.P(0,"DataSea.shiftInAll","resetting CS to "+Link.MED_CS+
                        " for node <"+node.Name+"> and <"+siblink.NodeR.Nam
                        e+">");
                if (siblink!=null)
                    siblink.set_CS(Link.MED_CS);
            }
        }
    } // end shiftInAll

    /**
    * mag_CSs_of_CNode magnify nodes connected to the Context Node of 'node'
    */
    static public Node mag_CSs_of_CNode (Node CNode)
    {
        int i, size;
        Link link=null;
        Node child=null, ret_node = null;

        CNode.more_mag();

        size = CNode.ContextLinks.size();
        //gui.P(0,"mag_CSs_of_CNode","Running on "+CNode.Name+" with "+size+" links.");
        for (i=0; i<size; i++) {
            link = (Link)(CNode.ContextLinks.elementAt(i));

            // Set CS to at least the DEFAULT for those related to this CNode
            if ((link.CS_R < Link.DEFAULT_CS) || (link.CS_L < Link.DEFAULT_CS))
                link.set_CS(Link.DEFAULT_CS);
            else
                link.more_CS();

            // Set mag of nodes linked by this link to at least MED MAG

```

05/23/00  
00:48:20

Confidential and Proprietary Property of Rocky Nevin  
DataSea.java

36

```
        if (link.NodeR.mag < Node.MED_MAG)
            link.NodeR.set_mag(Node.MED_MAG);
        else
            link.NodeR.more_mag();
        if (link.NodeL.mag < Node.MED_MAG)
            link.NodeL.set_mag(Node.MED_MAG);
        else
            link.NodeL.more_mag();

        //gui.P(0,"mag_CSs_of_CNode","Just increased mags and CS_R&L of link between <"+link.NodeR.Name+"> and <"+link.NodeL.Name+">");
    }

    //gui.P(0,"mag_CSs_of_CNode","Done.");
    return(ret_node);
} // end mag_CSs_of_CNode

/*
 * inh_CSs_of_CNode  inhibit nodes connected to the Context Node of 'node'
 */
static public Node inh_CSs_of_CNode (Node CNode)
{
    int i, size;
    Link link=null;
    Node child=null, ret_node = null;

    size = CNode.ContextLinks.size();
    gui.P(0,"inh_CSs_of_CNode","Running on <"+CNode.Name+"> with "+size+" links.");
    for (i=0; i<size; i++) {
        link = (Link)CNode.ContextLinks.elementAt(i);
        link.less_CS();
    }
    return(ret_node);
} // end inh_CSs_of_CNode

/*
 * set_CSs_of_CNode  set CSs of links connected to the Context Node of 'node'
 */
static public Node set_CSs_of_CNode (Node CNode, double CS)
{
    int i, size;
    Link link=null;
    Node child=null, ret_node = null;

    size = CNode.ContextLinks.size();
    for (i=0; i<size; i++) {
        link = (Link)CNode.ContextLinks.elementAt(i);
        link.set_CS(CS);
    }
    return(ret_node);
} // end set_CSs_of_CNode

/*
 * method local  local data nodes by making those of same type in a local visible
 */
public void chain (Node node) {
    int i, size;
    Node child;

    if (node == null) {
        gui.WARNING(0,"chain","No lastNode available");
        return;
    }
    if (node == null) {
        gui.WARNING(0,"local","No lastNode available");
        return;
    }
    node.set_mag(Node.MAX_MAG);
    gui.P(1,"local","Running on "+node.Name);

    size = node.Links.size();
    for (i=0; i<size; i++) {
        child = (Node)(node.getNodeAtLink(i));
        child.set_mag(child.mag + Node.DELTA_MAG);
        gui.P(1,"local","Running on "+child.Name);
    }
} // end local

/**
 ** method chain  chain data nodes by making those of same type in a chain visible
 */
public void chain (Node node) {
    int i, size;
    Node child;

    if (node == null) {
        gui.WARNING(0,"chain","No lastNode available");
        return;
    }
    if (node == null) {
        gui.WARNING(0,"local","No lastNode available");
        return;
    }
    node.set_mag(Node.MAX_MAG);
    gui.P(1,"local","Running on "+node.Name);

    size = node.Links.size();
    for (i=0; i<size; i++) {
        child = (Node)(node.getNodeAtLink(i));
        child.set_mag(child.mag + Node.DELTA_MAG);
        gui.P(1,"local","Running on "+child.Name);
    }
} // end local

/**
 ** method chain  chain data nodes by making those of same type in a chain visible
 */
public void chain (Node node) {
    int i, size;
    Node child;

    if (node == null) {
        gui.WARNING(0,"chain","No lastNode available");
        return;
    }
    if (node == null) {
        gui.WARNING(0,"local","No lastNode available");
        return;
    }
    node.set_mag(Node.MAX_MAG);
    gui.P(1,"local","Running on "+node.Name);

    size = node.Links.size();
    for (i=0; i<size; i++) {
        child = (Node)(node.getNodeAtLink(i));
        child.set_mag(child.mag + Node.DELTA_MAG);
        gui.P(1,"local","Running on "+child.Name);
    }
} // end local
```

05/23/00  
00:48:20

37

```
size = node.Links.size();
for (i=0; i<size; i++) {
    child = (Node)(node.getNodeAtLink(i));
    chain_r(child, child.Type, "distal");
    chain_r(child, child.Type, "proximal");
}
} // end chain

public void chain_r (Node node, String type, String direction) {
    Node child=null;
    int i, size;

    if (node == null)
        return;
    if (node == Root)
        return;

    gui.P(1, "chain_r", "Running on "+node.Name);

    // RECURSE
    size = node.Links.size();
    for (i=0; i<size; i++) {
        child = node.getNodeAtLink(i);
        // check for similar type
        if (((child.dist > node.dist) && direction.equals("distal") && child.Type.equalsIgnoreCase(type)) {
            chain_r(child, type, direction);
            gui.getToolkit().sync(); // drama
            gui.getToolkit().sync(); // drama
            gui.getToolkit().sync(); // drama
            child.more_mag(node);
        }
        else
            if (((child.dist < node.dist) && direction.equals("proximal") && child.Type.equalsIgnoreCase(type)) {
                chain_r(child, type, direction);
                gui.getToolkit().sync(); // drama
                gui.getToolkit().sync(); // drama
                gui.getToolkit().sync(); // drama
                child.more_mag(node);
            }
    }
    return;
} // end chain r

/**
 ** method pump pump up all distal nodes
 */
public void pump (Node node) {
    int i, size;
    Node child;

    if (node == null)
        node = Root;

    node.set_mag(Node.MAX_MAG-2);

    size = node.Links.size();
    for (i=0; i<size; i++) {
        child = (Node)(node.getNodeAtLink(i));
        if (((child.dist > node.dist) && (child.Type.equalsIgnoreCase(node.Type)))
            pump_r(node, child);
    }
} // end pump

/**
 ** method pump_r .pump data nodes by making them all visible
 */
public void pump_r (Node parent, Node node) {
    Node child=null;
    int i, size;

    if (node == null)
        return;
    if (node == Root)
        return;

    return;
    //if (node.isAN() || parent.isAN())
    //    return;

    node.set_mag(Node.MAX_MAG-2, parent); // allow use of CS from parent to node

    gui.P(1, "pump_r", "Running on "+node.Name+" from caller "+parent.Name);

    // RECURSE
    size = node.Links.size();
    for (i=0; i<size; i++) {
        child = node.getNodeAtLink(i);
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin DataSea.java

38

```
        if ((child.dist > node.dist) && (child.Type.equalsIgnoreCase(node.Type)))
            pump_r(node, child);
    }
    return;
} // end pump_r

/**
 ** method clean strip data nodes by stripping away DNs making them barely visible
 */
public void clean () {
    Node node;

    node = gui.lastNode; // try the lastNode first
    if (null == null) // try the Root next
        node = Root;

    System.out.println("clean(), node is "+node.Name);
    clean(node);
} // end clean

/**
 ** method clean strip data nodes by stripping away DNs making them barely visible
 */
public void clean (Node node) {
    if (node == null) {
        gui.WARNING(0, "clean", "Given node is null.");
        return;
    } else
        gui.P(0, "clean", "Running on "+node.Name);

    reset_mags(Node.BARELY_VISIBLE_MAG);
    mag_ans((Node)null, node, 2, 0);
    mag_ans((Node)null, node, 1, 0);
    node.set_mag(Node.MAX_MAG);
    //clean_r(node, count, distal(node));
} // end this version of clean

/**
 ** method clean strip data nodes by stripping away DNs making them barely visible
 */
public void clean (String[] words, int num_words) {
    Node node=null;

    if (num_words==1)
        node = gui.lastNode;
    else
        node = find_node_named(words[1]);

    clean(node);
} // end clean

/**
 ** method clean_r strip away data not's making them barely visible
 */
public void clean_r (Node node, int distal_count) {
    Node child=null;
    int i, size;
    int delay=0;

    if (node == null)
        return;

    // Scale how long we delay based on how many nodes we'll operate on
    // We want to take about 2 seconds for everything
    if (distal_count > 0)
        delay = 10000/distal_count;

    // RECURSE
    size = node.links.size();
    for (i=0; i<size; i++) {
        child = node.getNodeAtLink(i);
        if (child.dist > node.dist) {
            child.set_mag(Node.BARELY_VISIBLE_MAG);
            clean_r(child, distal_count);
        }
    }
    return;
} // end clean_r

/**
 ** mag_ans This mags AN's which are a certain number of AN links away 3/6/2000
 **
 */
public void mag_ans (Node caller, Node node, int target_level, int current_level) {
    int i, size;
    Node child=null;
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

DataSea.java

39

```
if (node == null)
    return;

//System.out.println("");
//System.out.println("["+current_level+"]");
if (caller != null)
    if (!caller.isAN && node.isAN)
        if (caller.goesUpstreamTo(node)) {
            //System.out.println("["+++upstream++]");
            if (current_level == target_level-1) {
                if (node.TS_diff() > 0) {
                    System.out.println("caller<"+caller.Name+">>set_mag<"+
                        +node.Name+">, current_level="+current_level);
                    if (node.mag < Node.MAX_MAG)
                        node.set_mag(Node.MAX_MAG);
                    else
                        node.more_mag();
                }
                current_level++;
            }
            else {
                //System.out.println("caller<"+caller.Name+"> not upstream -><"+node.Name
                +">");
                return;
            }
        }
        if (current_level >= target_level)
            return;

size = node.Links.size();
for (i=0; i<size; i++) {
    child = node.getNodeAtLink(i);
    if (child == Root) ;
    else
        if (child == caller) ;
    else
        if (child.dist > node.dist) { // is distal, no 'else'
            if (node.isAN) {
                if (!child.isAN)
                    ;
                else
                    { // child is AN also
                        if (node.goesUpstreamTo(child)) // no real 'else'
                            {
                                //System.out.println("upstream");
                                if (child.TS_diff() > 0) {
                                    if (current_level == target_level-1) {
                                        //System.out.println(" node<"+node.Name+">>SET_MAG<"+child.Name+">, current_level="+cu
                                        rent_level);
                                        System.out.println(" SET_MAG<"+child.Name+"
                                        >, "+current_level="+current_level);
                                        if (node.mag < Node.MAX_MAG)
                                            child.set_mag(Node.MAX_MAG);
                                        else
                                            child.more_mag();
                                    }
                                    if (current_level < target_level) // recurse
                                        mag_ans(node, child, target_level, 1+current_level);
                                }
                            }
                        }
                        else // node is not an AN, see if child is
                            mag_ans(node, child, target_level, current_level);
                    }
                return;
            } // end mag_ans

/**
** heavys This mags AN's which are a certain number of AN links away 3/6/2000
**
*/
public void heavys (Node caller, Node node, int target_level, int current_level) {
    int i, size;
    Node child=null;

    if (node == null)
        return;

    //System.out.println("");
    //System.out.println("["+current_level+"]");
    if (caller != null)
        if (!caller.isAN && node.isAN)
            if (caller.goesUpstreamTo(node)) {
                //System.out.println("["+++upstream++]");
                if (current_level == target_level-1) {
                    System.out.println("-MAG(+"+node.Links.size()+")<"+no
                    de.Name+">-");
                }
            }
        }
    }
}
```

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin

DataSea.java

40

```
node.set_mag(node.mag + node.links.size());
    }
    current_level++;
}

if (current_level >= target_level)
    return;

//if (caller != null)
//System.out.print("<" + caller.Name + ">" + "<" + node.Name + ">");
//else
//System.out.print("<NULL>" + ">" + node.Name + ">");

size = node.links.size();
for (i=0; i<size; i++) {
    child = node.getNodeAtLink(i);
    if (child == Root) ;
    else
        if (child == caller) ;
    else
        if (child.dist > node.dist) { // is distal, no 'else'
            //System.out.print("<" + child.Name + "> dist++ ");
            if (node.isAN) {
                if (!child.isAN)
                    else
                        { // child is AN also
                            if (node.goesUpstreamTo(child)) // no 'else'
                                {
                                    //System.out.print(" [upstream] ");
                                    if (current_level == target_level-1) {
                                        System.out.print(" MAG(+ " + node.links.size() + ")<" + node
                                            child.set_mag(node.mag + node.links.size());
                                    }
                                    if (current_level < target_level) // recurse
                                        heavans(node, child, target_level, 1+current_level)
                                }
                            //else
                                //System.out.print(" [downstream] ");
                        }
                    }
                else // node is not an AN, see if child is
                    heavans(node, child, target_level, current_level);
            }
        }
    //else
        }
```

```
        //System.out.print(" <" + child.Name + "> dist—— ");
    }
    return;
} // end heavans

/**
 ** enhance : ANs based on number of their LAN-children
 **
 */
public void enhance (String[] words, int num_words) {
    int i,j, size, node_size, child_counter, node_counter=0;
    Node node, child;
    String str=" ", Type="UNINITIALIZED TYPE";
    double new_mag;

    if (num_words == 1) {
        Type = "URL ";
        gui.P(0,"enhance", "Default Type is "+Type);
    }
    else {
        Type = words[1];
        gui.P(0,"enhance", "Type is "+Type);
    }

    size = node.vec.size();
    for (i=0; i<size; i++) {
        node = (Node)(node.vec.elementAt(i));
        if (node.Type.equalsIgnoreCase(Type)) {
            child_counter = 0;
            node_size = node.links.size();
            for (j=0; j<node.size; j++) { // NOW, COUNT CHILDREN OF CORRECT TYPE
                child = (Node)(node.getNodeAtLink(j));
                if (child.Type.equalsIgnoreCase(Type))
                    child_counter++;
            }
            if (child_counter>1) { // NOW, INCREASE THE MAG OF THE NODE 'child'
                new_mag = node.mag+0.3*child_counter;
                System.out.print(node.Name+" "+" +node.mag+" ">" +new_mag+" ");
                node.set_mag(new_mag);
                node_counter++;
            }
        }
    }
```

```
    }
```

```
gui.P.0,"enhance",node_counter+" nodes enhanced.");
```

```
return;
```

```
} // end enhance
```

```
/**
```

```
 ** choices list the ANs above BIG_MAG for each level of dist from Thes
```

```
*/
```

```
public void choices () {
```

```
int i,j, size, counter;
```

```
Node child;
```

```
String strj="";
```

```
gui.clear_global_str();
```

```
gui.add_to_global_str("Choices: ");
```

```
for (j=2;j<8;j++) {
```

```
    strj="";
```

```
    counter = 0;
```

```
    gui.add_to_global_str("Depth "+j+":");
```

```
    size = node_vec.size();
```

```
    for (i=0; i<size; i++) {
```

```
        child = (Node)(node_vec.elementAt(i));
```

```
        if (child.isAN && child.mag>Node.BIG_MAG && child.dist==j) {
```

```
            counter ++;
```

```
            strj=strj+"    "+child.Name+">";
```

```
        }
```

```
    }
```

```
    if (counter > 1)
```

```
        gui.add_to_global_str(strj);
```

```
    else
```

```
        if (counter == 1)
```

```
            gui.add_to_global_str("    (only 1-node)");
```

```
    else
```

```
        gui.add_to_global_str("    (empty)");
```

```
} // end choices
```

```
/**
```

```
 ** magtype mag directly linked nodes of same type with given direction.
```

```
*/
```

```
public void magtype (Node node, String type, char direction) {
```

```
int i, size;
```

```
Node child;
```

```
if (node==null)
```

```
    return;
```

```
if (type==null)
```

```
    type = node.Type;
```

```
if (direction=='')
```

```
    direction = 'd';
```

```
// ----- for children -----
```

```
size = node.Links.size();
```

```
for (i=0; i<size; i++) {
```

```
    child = (Node)(node.getNodeAtLink(i));
```

```
    if (child.Type.equalsIgnoreCase(type))
```

```
        if (((direction=='d') && (child.dist > node.dist))
```

```
            ||
```

```
            ((direction=='p') && (child.dist < node.dist)))
```

```
        {
```

```
            child.more_mag(node);
```

```
            magtype(child, type, direction);
```

```
        }
```

```
// ----- end for children -----
```

```
return;
```

```
}
```

```
/**
```

```
 ** magall
```

```
 **
```

```
*/
```

```
public void magall (Node node) {
```

```
mag.r(null, node, "distal", gui.INFINITE_DEPTH, 0, "+", true);
```

```
mag.r(null, node, "distal", gui.INFINITE_DEPTH, 0, "+", true);
```

```
} // end magall
```

```
/**
 ** magd
 **
 */
public void magd () {

    if (gui.lastNode != null)
        mgr.r(null, gui.lastNode, "downstream", gui.INFINITE_DEPTH, 0, "+", true);
} // end magd

/*
/**
 ** inhd
 **
 */
public void inhd () {

    if (gui.lastNode != null)
        mgr.r(null, gui.lastNode, "downstream", gui.INFINITE_DEPTH, 0, "-", true);
} // end inhd

/*
 * mag Full version
 */
public void mag (String[] words, int num_words, String direction, int max_dist, String more_or_less)
{
    Node node=null;

    if (num_words==1)
        node = gui.lastNode;
    else
        node = find_node_named(words[1]);

    if (node != null) {
        mgr.r((Node)null, node, direction, max_dist, 0, more_or_less, false);
    }
    else
    {
        GUI.WARNING(0, "DataSea.mag", "Can't get a node; num_words="+num_words);
    }
} //end mae

/**
 * mag simplest version
 */
public void mag (Node node) {
    mgr.r((Node)null, node, "distal", 5, 0, "+", false);
} // end mag

/*
 * mag simplest version more_or_less is either "+" or "-"
 */
public void mag (Node node, String direction, String more_or_less) {
    mgr.r((Node)null, node, direction, 5, 0, more_or_less, false);
} // end mag (simplest)

/*
 * mag simplest version more_or_less is either "+" or "-"
 */
public void mag (Node node, String direction, String more_or_less, int dist) {
    mgr.r((Node)null, node, direction, dist, 0, more_or_less, false);
} // end mag (simplest)

/**
 ** method mgr 7-variable version
 */
public void mgr (Node node, Node child, String direction, int max_call_depth, int this_call_depth,
String more_or_less, boolean cross, AN_DN_boundary) {
    Node grand_child=null;
    int i, size;
    boolean distal=false, proximal=false, both=false, OK=false;
    boolean downstream=false, upstream=false;

    if (child == null)
        return;
    if ((node != null) && (child == Root)) // OK to mag root if there's no caller
        return;

    if (child.isCN) {
        if (gui.drawCN) {
            if (more_or_less.equalsIgnoreCase("+"))
                mag_CSs_of_CNnode(child);
            else

```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

DataSea.java

43

```
        inh_CSs_of_CNode(child);
    }
}

// HANDLE MODE
// SET UP THE MODES
if (direction.equalsIgnoreCase("downstream"))
    downstream = true;
if (direction.equalsIgnoreCase("upstream"))
    upstream = true;
if (direction.equalsIgnoreCase("distal"))
    distal = true;
if (direction.equalsIgnoreCase("proximal"))
    proximal = true;
if (direction.equalsIgnoreCase("both"))
    both = true;

// INCREMENT this.call_depth refers to number of recursive calls, not Node.dist
if (++this.call_depth > max_call_depth)
    return;

//System.err.println("mag.r: "+this.call_depth+" < "+max_call_depth);

// HANDLE THIS_DIST
// INCREASE the mag of the child arg'
if (more_or_less.equalsIgnoreCase("+")) {
    if (child_TS.diff() > 0) { // this TS is before GUI.lastCommandTS
        //System.err.println("mag.r: depths: "+this.call_depth+" < "+max_call_depth+">, more_mag("+child.
        Name+"")");
        child.more_mag(node);
    }
}
else {
    child.less_mag(node);
}

// RECURSE
size = child.Links.size();
for (i=0; i<size; i++) {
    grand_child = child.getNodeAtLink(i);
    if (proximal && (grand_child.dist < child.dist))
        OK = true;

    else
        if (both)
            OK = true;
        else
            if (distal && (grand_child.dist > child.dist))
                OK = true;
            else
                if (downstream && child.getPol(child)=='+')
                    OK = true;
                else
                    if (upstream && child.getPol(child)=='-')
                        OK = true;

    // sense AN-DN transition:
    // Don't recurse by passing through AN to DN: DN's refer to other contexts
    // dist=2 means that child is connected to starting point
    if (node != null)
        if ((this.call_depth >= 2) && (child.isAN && !grand_child.isAN) && !cross_A
            N_DN_boundary) {
            OK = false;
        }
        if (grand_child.StopsSpread)
            OK = false;
        if (OK) {
            if (distal || both || downstream)
                mag_r(child, grand_child, "distal", max_call_depth, this.call_depth, mo
                re_or_less, cross_AN_DN_boundary);
            if (proximal || both || upstream)
                mag_r(child, grand_child, "proximal", max_call_depth, this.call_depth,
                more_or_less, cross_AN_DN_boundary);
            OK = false;
        }
    }
    return;
} // end mag_r

// CLEAN UP FUNCTIONS.
/**
** simplify hide DNs distal to ANs
**
*/
public void simplify (String[] words, int num_words) {
    int i, size;
    Node node, in;
```

## 44

```

        simplify_recursive(child, tn);
    }

    } // end simplify_recursive

/**
 ** strip_r marginalize all non-AN's after first AN distal to node
 ** */
public void strip_r (Node parent, Node node) {
    int i, size;
    Node child;
    boolean inhibit_it = false;

    if (node == null)
        return;

    size = node.links.size();
    for (i=0; i<size; i++) {
        child = (Node)(node.getLink(i));
        if (child.dist > node.dist) {
            if (parent != null)
                if (parent.isAN && !child.isAN) // check AN-!AN boundary
                    inhibit_it = true;
            if (inhibit_it)
                inhibit(child);
            else
                strip_r(node, child);
        }
    }
} // end strip_r

/**
 ** strip marginalize all non-AN's after first AN distal to node
 ** */
public void strip (String[] words, int num_words) {
    Node node=null;

    if (num_words==1)
        node = gui.lastNode;
    else
        node = find_node_named(words[1]);
}

```

```
strip,r((Node)null, node);  
} // end strip
```

```
/**  
** inhibit put below threshold the distal nodes, and the proximal nodes up to first AN  
** this version handles words  
*/
```

```
public void inhibit (String[] words, int num_words) {  
    int i,j, size, size_j;  
    Node node, child, tn;
```

```
// HANDLE IDENTIFYING THE CORRECT NODE TO START ON  
    if (num_words==1)
```

```
        node = gui.lastNode;  
    else  
        node = find_node,named(words[1]);
```

```
// CHECK FOR ERRORS
```

```
    if (node==null) {  
        if (num_words>1)
```

```
        GUI.WARNING(0,"DataSea.inhibit","Can't find node "+words[1]);
```

```
    else
```

```
        GUI.WARNING(0,"DataSea.inhibit","Neither Name given nor existing lastNode.");
```

```
        return;
```

```
    }
```

```
    else
```

```
        GUI.P(0,"inhibit","Found node named '"+node.Name+"'");
```

```
//
```

```
gui.show_node_once(node);
```

```
inhibit(node);
```

```
} // end inhibit
```

```
/*
```

```
** inhibit single arg version, inhibit proximally and distally from node
```

```
**
```

```
*/
```

```
public void inhibit (Node node) {
```

```
    inhibit_distally(node);
```

```
    // inhibit_proximally(node);
```

```
} // end inhibit
```

```
/*
```

```
** inhibit_distally put below threshold the distal nodes, and the proximal nodes up to first AN
```

```
**
```

```
*/
```

```
public void inhibit_distally (Node node) {
```

```
    int i, size;
```

```
    Node child=null;
```

```
    node.less_mag();
```

```
    GUI.shrinking_allowed = false;
```

```
    size = node.Links.size();
```

```
    for (i=0; i<size; i++) {
```

```
        child = (Node)(node.getNodeAtLink(i));
```

```
        if (child.dist > node.dist)
```

```
            inhibit_distally(child); // decrease all distal nodes
```

```
    }
```

```
    GUI.shrinking_allowed = true;
```

```
    return;
```

```
} // end inhibit_distally
```

```
/*
```

```
** inhibit_proximally reduce proximal nodes up to first AN
```

```
**
```

```
*/
```

```
public void inhibit_proximally (Node node) {
```

```
    int i, size;
```

```
    Node parent=null;
```

```
    if (null == POV) // go backwards and if AN is found, decrease distal from it
```

```
        return;
```

```
    GUI.P(0,"inhibit_proximally", "node="+node.Name);
```

```
    size = node.Links.size();
```

```
    for (i=0; i<size; i++) {
```

```
        parent = (Node)(node.getNodeAtLink(i));
```

```
        if (parent.dist < node.dist) {
```

```
            if (parent.isAN && (parent.dist <= 2)) {
```

```
                GUI.P(0,"inhibit_proximally", "breaking for AN parent="+parent.Name+" dist="+parent
```

```
                .dist);
```

```
                break;
```

```
            }
```

```
            if (parent.dist > 2) {
```

```
                parent.less_mag();
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

DataSea.java

46

```
// RECURSE
GUI.shrinking_allowed = false;
inhibit_proximally(parent);
GUI.shrinking_allowed = true;
}
} // if parent's dist < node's

} // end inhibit_proximally

/**
 ** vote_branch
 ** spread mag changes (+ or -, depending on String s)
 ** both proximally and distally for positive, distal only for negative
 */
public void vote_branch (String[] words, int num_words, String s) {
    Node node = null;

    if (num_words==1)
        node = gui.lastNode;
    else
        node = find_node_named(words[1]);

// CHECK FOR ERRORS
    if (node==null) {
        if (num_words>1)
            GUI.WARNING(0,"vote_branch","Can't find node "+words[1]);
        return;
    }
    else
        vote_branch(node, s);
} // end vote_branch

int i;
public void vote_branch (Node node, String s) {
    String saved_spread_mode = gui.mode_obj.spread_mode;

    if (node == null)
        node = gui.lastNode;
    if (node==null)
        return;

    GUI.P(0,"vote_branch":"Active "+"s+" . on Node="+node.Name);

    if (s.equals("+")) { // PLUS
        gui.mode_obj.spread_mode = "both";
        mag(node, "both", "s+");
    }
    if (s.equals("-")) { // MINUS
        gui.mode_obj.spread_mode = "distal";
        GUI.shrinking_allowed = false;
        strip_r((Node)null, node);
        node.set_mag(gui.text_threshold - 0.1);
        GUI.shrinking_allowed = true;
    }
    gui.mode_obj.spread_mode = saved_spr_ad_mode;
    return;
} // end vote_branch

/**
 ** method sim amplify similar nodes to target (same type, linked ANs)
 ** if started on AN, call on non-AN children
 */
public void sim (String[] words, int num_words, char sign) {
    int i, size;
    Node node, child;

// HANDLE IDENTIFYING THE CORRECT NODE TO START ON
    if (num_words==1)
        node = gui.lastNode;
    else
        node = find_node_named(words[1]);

// CHECK FOR ERRORS
    if (node==null) {
        if (num_words>1)
            GUI.WARNING(0,"DataSea.sim"+sign,"Can't find node "+words[1]);
        else
            GUI.WARNING(0,"DataSea.sim"+sign,"Neither Name given nor existing lastNode.");
        return;
    }
    else
        GUI.P(0,"sim"+sign,"Found node named "+node.Name+"");

    if (node.isURL) {
        spread_url_sim(node, sign);
    }
}
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

DataSea.java

47

```
if (node.isAN) { // call on all children, no recursion really
    size = node.Links.size();
    for (i=0; i<size; i++) {
        child = node.getNodeAtLink(i);
        if (!child.isAN)
            sim(child, sign);
    }
    return;
}
else
    sim(node, sign);

node.set_mag(Node.MAX_MAG); // Make sure we don't normalize this one down

return;
} // end sim

/**
 ** method sim    amplify similar nodes to target (same type, linked ANs)
 **               Looks only at child of directly connected ANs, of same type as 'nod
 **               e'
 */
public void sim (Node node, char sign) {
    int i, j, i_size, j_size;
    Node child, grand_child;

// HANDLE IDENTIFYING THE CORRECT NODE TO START ON
// -----

    spread_url_sim(node, sign);

    i_size = node.Links.size();
    for (i=0; i<i_size; i++) {
        child = node.getNodeAtLink(i);
        if (child.isAN)
        {
            GUI.P(0, "sim"+sign, "Type=AN, I_size="+i_size+", "+node.Name+"["+i+"] is "+child.N
            ame);
            j_size = child.Links.size();
            for (j=0; j<j_size; j++) {
                grand_child = child.getNodeAtLink(j);

// DON'T MAG THE AN's ... only the non-ANs
// -----
                if (sign == '+')
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

DataSea.java

48

```
        child.less_mag();
        if ('-' == node.getPol(child)) {
            j.size = child.Links.size();
            for (j=0; j<j.size; j++) {
                grand_child = child.getNodeAtLink(j);
                if (grand_child.isURL) {
                    if ('+' == child.getPol(grand_child)) {
                        GUI.P(0,"spread_url_sim"+sign,
                            "2nd-level:"+child.Name+"("+child.Type+")->"+grand_c
                                hild.Name+"("
                                    +grand_child.Type+") Pol="
                                    +child.getPol(grand_child);
                                    if (sign == '+')
                                        grand_child.more_mag();
                                    else
                                        grand_child.less_mag();
                                }
                            }
                        }
                    }
                }
            }
        } // end spread_url_sim

        /**
         ** tickle
         **
         public void tickle (Node node) {
             int i, size;
             Node child;

             size = node.Links.size();
             for (i=0; i<size; i++) {
                 child = node.getNodeAtLink(i);
                 if (child.isURL) {
                     GUI.P(0,"tickle ", node.Name+" tickling " +child.Name);
                     child.more_mag();
                 }
             }
         } // end tickle
         */

        /**
         ** method unsim      decrease similar nodes to target (same vpc, linked ANs)
         */
    }

    public void unsim (String[] words, int num_words) {
        int i, j, i_size, j_size;
        Node node, child, grand_child;

        // HANDLE IDENTIFYING THE CORRECT NODE TO START ON
        if (num_words==1)
            node = gui.lastNode;
        else
            node = find_node_named(words[1]);

        // CHECK FOR ERRORS
        if (node==null) {
            if (num_words>1)
                GUI.WARNING(0,"DataSea unsim ", "Can't find node "+words[1]);
            else
                GUI.WARNING(0,"DataSea unsim ", "Neither Name given nor existing lastNode.");
            return;
        }
        else
            GUI.P(0,"unsim ", "Found node named "+node.Name+"");

        GUI.shrinking_allowed = false;

        gui.show_node_once(node);
        i_size = node.Links.size();
        for (i=0; i<i_size; i++) {
            child = node.getNodeAtLink(i);
            if (child.isAN && child!=node) {
                j_size = child.Links.size();
                for (j=0; j<j_size; j++) {
                    grand_child = child.getNodeAtLink(j);
                    if (grand_child.Type.equals(node.Type) && grand_child!=node)
                        grand_child.less_mag(child); // de-emphasize ANs
                }
            }
        }

        GUI.shrinking_allowed = true;

        return;
    } // end unsim
    */

    /**
     ** isolate
     **
    }
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

DataSea.java

49

```

*/
public void isolate () {
    int size, links_size, i, j;
    Node node;
    Link link=null;
    double max_CS=0;

    size = node_vec.size();
    GUI.P(0,"isolate","Scanning <"+size+"> nodes.");
    for (i=0; i<size; i++) {
        node = (Node)node_vec.elementAt(i);
        // DETERMINE THE MAXIMUM CS FROM ALL THE LINKS TO THIS NODE
        max_CS = get_max_CS(node);
        // NOW SET THE MAG PROPORTIONAL TO THE MAXIMUM CS TO THIS NODE
        System.out.println(node.Name+"<"+gui_prec(max_CS,3)+">");
        if (max_CS < 0.9*Link.DEFAULT_CS)
            node.set_mag(max_CS/Link.DEFAULT_CS);
    }
    // end isolate

    /**
     ** get_max_CS
     **
     */
    public double get_max_CS (Node node) {
        int i, size;
        Link link;
        double max_CS=0;

        size = node.links.size();
        for (i=0; i<size; i++) {
            link = (Link)node.links.elementAt(i);
            if (max_CS < link_CS_R)
                max_CS = link_CS_R;
            if (max_CS < link_CS_L)
                max_CS = link_CS_L;
        }

        return(max_CS);
    } // end get_max_CS

    /**
     ** halve_mags
     */
    public void halve_mags () {
        int size, i;
        Node tn;

        size = node_vec.size();
        GUI.P(0,"halve_mags","Halving all mags ... size="+size);
        for (i=0; i<size; i++) {
            tn = (Node)node_vec.elementAt(i);
            tn.set_mag(tn.mag * 0.5);
        }
    } // end halve_mags

    /**
     ** drill_kernel print list of distal ANs
     **
     */
    public void drill_kernel (Node caller) {
        int i, size;
        Node child;

        size = caller.links.size();
        for (i=0; i<size; i++) {
            child = (Node)(caller.getNodeAtLink(i));
            if (child.dist > caller.dist) {
                if (caller.isDN && child.isAN) { // dim' distal DNs after ANs
                    gui_global.strf[0] = gui_global.strf[0] + " "+child.Name+"="+caller.Name;
                }
                else if (caller.isDN && child.isDN)
                    drill_kernel(child);
            }
        }
    } // end drill_kernel

    /**
     ** drill
     **
     */
    public void drill (String[] words, int num_words) {
        int i, size;
        Node node, tn;
    }

```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

DataSea.java

50

```
// HANDLE IDENTIFYING THE CORRECT NODE TO START ON
if (num_words==1)
    node = gui.lastNode;
else
    node = find_node_named(words[1]);

// CHECK FOR ERRORS
if (node==null) {
    if (num_words>1)
        GUI.WARNING(0,"DataSea.drill","Can't find node "+words[1]);
    else
        GUI.WARNING(0,"DataSea.drill","Neither Name given nor existing lastNode.");
    return;
}
else
    GUI.P(0,"drill","Found node named "+node.Name+"");

// -----
gui.show_node_once(node);
size = node.Links.size();

gui.global_strf(0) = "Drilling down "+node.Name+"",
drill_kernel(node);
gui.global_strf(0) = gui.global_strf(0) + " ";
gui.global_str_size = 1;

GUI.P(0,"drill",gui.global_strf(0));

} // end drill

/**
 ** test
 **
 */
public void test () {
    int i, size;
    Node tn;

    gui.test();

    gui.show_node_once(find_node_named("Bob"));
    gui.sleep(500);
    gui.show_node_once(find_node_named("phone"));
    gui.sleep(500);

    gui.show_node_once(find_node_named("Web"));
    gui.sleep(500);
    gui.show_node_once(find_node_named("Directory"));
    gui.sleep(500);

} // end test

/**
 ** set_POV
 **
 */
public void set_POV () {
    int i, size;
    Node tn;

    if (GUI.lastNode != null) {
        POV = GUI.lastNode;
    }

} // end set_POV

/**
 ** repeat_priorCommand
 **
 */
public void repeat_priorCommand () {
    if (gui.priorCommand != null) {
        GUI.P(0,"repeat_priorCommand","repeating "+gui.priorCommand);
        gui.input_string_input(gui.priorCommand);
    }
    return;
} // end repeat_priorCommand

/**
 ** reset_mags
 **
 */
public void reset_mags () {
    reset_mags(Node.DEFAULT_MAG);
    return;
} // end reset_mags
```

```

/**
 ** reset_mags
 **
 */
    public void reset_mags (double newmag) {
        int size, i, j, child_size;
        Node node, child;
        double max_CS=0;

        size = node_vec.size();
        gui_text.threshold = gui.DEFAULT_TEXT_THRESHOLD;
        Root.set_mag(Node.MAX_MAG); // to keep others from being normalized to MAX_MAG
        for (i=0; i<size; i++) {
            node = (Node)(node_vec.elementAt(i));
            max_CS = get_max_CS(node);
            node.set_mag(newmag*max_CS); // 12/24/99 added max_CS
        }
    } // end reset_mags

/**
 ** reset_ANmags.no_history
 **
 */
    public void reset_ANmags.no_history (double newmag) {
        int size, i, j, child_size;
        Node node, child;
        double max_CS=0;

        size = node_vec.size();
        gui_text.threshold = gui.DEFAULT_TEXT_THRESHOLD;
        Root.set_mag(Node.MAX_MAG); // to keep others from being normalized to MAX_MAG
        for (i=0; i<size; i++) {
            node = (Node)(node_vec.elementAt(i));
            if (node.isAN())
                node.set_mag.no_history(newmag);
        }
    } // end reset_ANmags.no_history

/**
 ** reset_CSs
 **
 */
    public void reset_CSs () {
        int size, j_size, i, j;
        Node node;
        Link siblink;

        size = node_vec.size();
        for (i=0; i<size; i++) {
            node = (Node)(node_vec.elementAt(i));
            node.min_mag = Node.MIN_MAG;
            j_size = node.links.size();
            for (j=0; j<j_size; j++) {
                siblink = node.getLink(j);
                if (i<10)
                    GULP(0, "DataSea.reset_CSs ", resetting CS to "+Link.DEFAULT_CS
                        " for node <"+node.Name+"> and <"+siblink.NodeR.Name
                        e+">");
                if (siblink!=null) {
                    siblink.set_CS(Link.DEFAULT_CS);
                }
            }
        } // end reset_CSs

/**
 ** self_mag
 **
 */
    public void self_mag (Node node) {
        int i, j, size, child_size;
        Node child, grand_child;
        double CS, max_CS;

        size = node_vec.size();
        for (i=0; i<size; i++) {
            child = (Node)(node_vec.elementAt(i));
            child_size = child.links.size();
            max_CS = 0;
            for (j=0; j<child_size; j++) {
                grand_child = child.getLink(j);
                CS = child.get_CS(grand_child);
                if (CS > max_CS)
                    max_CS = CS;
            }
            for (j=0; j<child_size; j++) {
                grand_child = child.getLink(j);
                if (CS > max_CS)
                    max_CS = CS;
            }
        }
    }

```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

## DataSea.java

52

```

    }

    } // end self_mag
    */

}

/**
 ** recent
 **
 */
public void recent () {
    int i, size;
    long min_TS, max_TS, median_TS;
    Node tn;
    Node bob, files;

    min_TS = GUI.current_TS;
    max_TS = 0;

    size = node_vec.size();
    /*****
    for (i=0; i<size; i++) {
        tn = (Node)(node_vec.elementAt(i));
        if (tn.created_TS < min_TS)
            min_TS = tn.created_TS;
        if (tn.created_TS > max_TS)
            max_TS = tn.created_TS;
    } // now we have min and max

    median_TS = min_TS + (max_TS - min_TS)/2;
    *****/

    for (i=0; i<size; i++) {
        tn = (Node)(node_vec.elementAt(i));
        if (tn.created_TS > GUI.current_TS-30000)
            tn.set_mag(tn_mag * 2);
    }

    /*****
    GUI.P(0,"recent", "min="+min_TS+", max="+max_TS+", median="+median_TS);
    bob = find_node_named("Bob");
    if (bob != null)
        GUI.P(0,"recent", "Bob created TS="+bob.created_TS);
    *****/
}

files = find_node_named("Files");
if (files != null)
    GUI.P(0,"recent", "Files created TS="+files.created_TS);
    *****/

} // end recent

/**
 ** word_fn run a function on selected_node(s)
 **
 */
public void word_fn (String[] words, int num_words) {
    Node node, inode;
    Node node_array[];
    int i, j, size;

    GUI.P(0,"DataSea word_fn", "Started.");

    node_array = new Node[2];

    node_array[0] = (Node)(gui.selected_nodes_vec.elementAt(0));
    node_array[1] = (Node)(gui.selected_nodes_vec.elementAt(1));
    if (node_array[0]==null)
    {
        GUI.ERROR(0,"word_fn", "node_array[0] is null");
        return;
    }
    if (node_array[1]==null) {
        GUI.ERROR(0,"word_fn", "node_array[1] is null");
        return;
    }

    GUI.P(0,"fn:", "node_array[0].Name+" "+node_array[1].Name);
    GUI.P(0,"fn:", (node_array[0].getNodeAtLink(1)).Name+" "+(node_array[1].getNodeAtLink(1)).Name);
    GUI.P(0,"fn:", (node_array[0].getNodeAtLink(2)).Name+" "+(node_array[1].getNodeAtLink(2)).Name);
    GUI.P(0,"fn:", (node_array[0].getNodeAtLink(3)).Name+" "+(node_array[1].getNodeAtLink(3)).Name);

    /*****
    for (j=0; j<2; j++) {
        node = node_array[j];
        size = node.links.size();
        GUI.P(0,"word_fn", "node.Name+" size is "+size);
        for (i=0; i<size; i++)

```



-249-

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin

DataSea.java

04

```
if (AN2==null) {
    GUI.ERROR(0,"SS", "AN2 is null");
    return;
}
DN1=null;
DN2=null;

// NOW WE HAVE AN1 and AN2 TO OPERATE FROM
SS_node = new Node("SSheet", "Form", "Spread Sheet node", 100, 100, 120, 140);

// Run through AN's, get all DN's
gui.global.str[gui.global.str.size++] = AN1.Name+" "+AN2.Name;
gui.global.str[gui.global.str.size++] = "-----";
i=0; // initialize counter to catch all DN's of the AN's we have established
DN1=AN1.getDN(i);
if (DN1 != null)
    DN2=DN1.getDN_connected_to_AN(AN2);
else
    DN2=null;
    SS_node.link(DN1);
    SS_node.link(DN2);
    set_child_position(SS_node, DN1, 0.01, 0.1*(++line_num));
    set_child_position(SS_node, DN2, 0.51, 0.1*(line_num));

while (DN2!=null && gui.global.str.size<9)
{
    gui.global.str[gui.global.str.size++] = DN1.Name+" "+DN2.Name;
    i++;
    DN1=AN1.getDN(i);
    if (DN1 != null)
        DN2=DN1.getDN_connected_to_AN(AN2);
    else
        DN2=null;
    if ((DN1 != null) && (DN2 != null)) {
        SS_node.link(DN1);
        SS_node.link(DN2);
        set_child_position(SS_node, DN1, 0.01, 0.1*(++line_num));
        set_child_position(SS_node, DN2, 0.51, 0.1*(line_num));
    }
}
line_num++;
SS_node.link(AN1);
SS_node.link(AN2);
set_child_position(SS_node, AN1, 0.01, 0.1*(++line_num));
set_child_position(SS_node, AN2, 0.51, 0.1*(line_num));

reset_and_zoom("SSheet");
GUI.P(0,"SS", "Done.");
return;
} // end SS

/**
** word_mail create a 'mail' application DEFUNCT
*/
public void word_mail (String[] words, int num_words) { // DEFUNCT
    Node name_node=null, address_node=null, target_node=null;
    Node MailForm_node=null, To_node=null;

    GUI.P(0,"DataSea.word_mail", "Started.");

    if (num_words==1) {
        target_node = gui.lastNode;
    }
    else if (num_words==2) {
        target_node = find_node_named(words[1]);
    }
    if (target_node == null) {
        GUI.ERROR(0,"DataSea.word_mail", "No node selected nor given as name node.");
        return;
    }
    set_dist_start(target_node); // NEED FOR PROPAGATING

// FIND A NAME
GUI.P(0,"DataSea.word_mail", "Looking for 'name'");
name_node=gui.lastNode.getAN_named("name", 4);
if (name_node == null) {
    GUI.ERROR(0,"DataSea.word_mail", "No node found for name node, given target_node");
    return;
}
return;
}

// FIND AN ADDRESS
GUI.P(0,"DataSea.word_mail", "Looking for 'address'");
address_node=gui.lastNode.getAN_named("address", 4);
```

05/23/00  
00:48:20

55

```
        if (address_node == null) {
            GUI.ERROR(0, "DataSea.word_mail", "No node found for address node, given
target_node ""
            +target_node.Name+"");
            return;
        }

DataSea.needsUpdate = true; // Reset all Node.dist's

GUI.P(0, "DataSea.word_mail", "Creating mail form for target node ""+target_node.Name+"");
MailForm node = new Node("MailForm", "Form", "", 50, 50, 50, 50);
To_node = new Node("To: ", "DN", "", -20, 10, 1, 1);
name_node.setX("word_mail", 30, 10);
address_node.setX("word_mail", 30, 30);
MailForm_node.link(address_node);
MailForm_node.link(name_node);
MailForm_node.link(To_node);

    } // end word_mail DEFUNCT

/**
 ** word_focus make node the POV
 */
    public void word_focus (String[] words, int num_words) {
        Node node;

        GUI.P(0, "DataSea.word_focus", "Started.");
        // HANDLE IDENTIFYING THE CORRECT NODE TO START ON
        if (num_words==1)
            node = gui.lastNode;
        else
            node = find_node.named(words[1]);

        // CHECK FOR ERRORS
        if (node==null)
        {
            GUI.ERROR(0, "DataSea.word_focus", "Can't find node ""+words[1]);
            return;
        }
        // NOW, DO WHAT'S ASKED OF US
        POV=node;
    } // end word_focus
```

```
/**
 ** find_max Select for us the node with greatest mag
 **
 */
    public void find_max () {
        int i, size;
        Node node=null, saved_node=null;
        double saved_mag=Node.MIN_MAG;

        size = node_vec.size();
        for (i=0; i<size; i++) {
            node = (Node)(node_vec.elementAt(i));
            if (node.mag > saved_mag) {
                saved_node = node;
                saved_mag = node.mag;
            }
        }
        // Now, select for us the node with greatest mag.
        gui.lastNode = saved_node;
        if (saved_node != null)
            GUI.P(0, "find_max", "max mag is ""+saved_node.mag+" for ""+saved_node.Name);
        else
            GUI.P(0, "find_max", "No saved node found.");
        return;
    } // end find_max

/**
 ** pulse From all AN's, add AN_mag to children.
 ** Used after drawDN was false and not positioned
 **
 */
    public void pulse () {
        int i, size, j, j_size;
        Node node=null, child=null;

        GUI.P(0, "DataSea.pulse", "Started.");
        GUI.P(0, "DataSea.pulse", "Started.");

        size = node_vec.size();
        for (i=0; i<size; i++) {
            node = (Node)(node_vec.elementAt(i));
```

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin

DataSea.java

56

```
j.size = node.Links.size();
for (j=0; j<j.size; j++) {
    child = (Node)(node.getNodeAtLink(j));
    if (child.isDN) {
        child.set_mag(child.mag + node.mag);
    }
}

GUI.P(0,"DataSea.pulse","Done.");

return;
} // end pulse

/**
 ** word_select select node 'find' -> set lastNode, 'select' -> set node.isSelected
 */
public void word_select (String[] words, int num_words) {
    Node node=null;
    int i, size;

    String without_command="";
    if (num_words >= 2) {
        string_without_command = words[1];
        for (i=2; i< num_words; i++)
            string_without_command = string_without_command+" "+words[i];
    }

    GUI.P(1,"DataSea.word_select","Started.");
    // HANDLE IDENTIFYING THE CORRECT NODE TO START ON
    if (num_words==1)
        node = gui.lastNode;
    else {
        if (words[0].equalsIgnoreCase("AN"))
            node = find_node_named(words[1], "AN");
        else
            if (words[0].equalsIgnoreCase("DN"))
                node = find_node_named(words[1], "DN");
            else
                node = find_node_named(string_without_command);
    }

    // CHECK FOR ERRORS

    if (node==null)
    {
        GUI.WARNING(0,"DataSea.word_select","Can't find "+string_without_command);
        return;
    }
    // NOW, DO WHAT'S ASKED OF US
    gui.lastNode = node;
    gui.show_node_once(node);

    if (words[0].equalsIgnoreCase("select")) {
        node.isSelected = true;
        gui.selected_nodes_vec.addElement(node);
    }

    } // end select

    /**
     ** word_unselect unselect all nodes
     */
    public void word_unselect (String[] words, int num_words) {
        Node node, child;
        int i, size;
        GUI.P(0,"DataSea.word_unselect","Unselecting all nodes.");

        if (words[0].equalsIgnoreCase("unselect")) // words[0].equalsIgnoreCase("un") {
            if (num_words == 2) {
                node = find_node_named(words[1]);
                if (node != null) {
                    unselect(node);
                }
            } else {
                unselect((Node)null);
            }
        } // end check of 'unselect' command
        return;
    } // end word_unselect

    /**
     ** unselect unselect
     */
    public void unselect (Node node) {
        Node child;
```

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin

DataSea.java

57

```
int i, size;
GUI.P(0,"DataSea.unselect","Unselecting all nodes.");

// Loop on all nodes, unselect them and remove from list
size = node_vec.size();
for (i=0; i<size; i++) {
    child = (Node)(node_vec.elementAt(i));
    child.isSelected = false;
    gui.selected_nodes_vec.removeElement(child);
}
return;
} // end unselect

/**
 ** word.delete    delete node  WARNING!  dangerous method
 */
public void word_delete (String[] words, int num_words) {
    Node node;

    GUI.P(0,"DataSea.word_delete","Started.");
    // HANDLE IDENTIFYING THE CORRECT NODE TO START ON
    if (num_words==1)
        node = gui.lastNode;
    else
        node = find_node_named(words[1]);

    // CHECK FOR ERRORS
    if (node==null)
    {
        GUI.WARNING(0,"DataSea.word_delete","Can't find node "+words[1]);
        return;
    }

    // NOW, DO WHAT'S ASKED OF US
    node.unlink_all();

    // end word_delete (delete node);

/**
 ** word.release
 */
public void word_release (String[] words, int num_words) {
    Node node=null;

    String name="UNKNOWN";
    GUI.P(0,"DataSea.word_release","Started.");

    if (POV == null) {
        GUI.WARNING(0,"DataSea.word_release","Need a POV.");
        return;
    }

    // HANDLE IDENTIFYING THE CORRECT NODE TO START ON
    if (num_words==1)
        node = gui.lastNode;
    else
        if (num_words==2) {
            node = find_node_named(words[1]);
            if (node != null)
                name = node.Name;
        }

    // CHECK FOR ERRORS
    if (node==null)
    {
        GUI.WARNING(0,"DataSea.word_release","Can't find node <"+name+">");
        return;
    }
    else
        name = node.Name;

    GUI.P(0,"DataSea.word_release","About to release '"+node.Name+"' from POV");
    // NOW, DO WHAT'S ASKED OF US
    POV.unlink(node);
    node.unlink(POV);
    return;

} // end word_release

/**
 ** word.unlink
 */
public void word_unlink (String[] words, int num_words) {
    Node node1=null, node2=null;

    GUI.P(0,"DataSea.word_unlink","Started.");
    if (num_words<2) {
```

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin

DataSea.java

58

```
GUI.ERROR(0,"DataSea.word_unlink","Need at least one argument (unlink[link_between]
node1 node2)");
return;
    }
    if (num_words>3) {
        GUI.ERROR(0,"DataSea.word_unlink","Need at most two arguments (unlink[link_between]
node1 node2)");
        return;
    }

// HANDLE IDENTIFYING THE CORRECT NODE TO START ON
if (num_words==2)
{
    if (POV != null)
        node1 = POV;
    else
        node1 = gui.lastNode;
    node2 = find_node_named(words[1]);
}
if (num_words==3)
{
    node1 = find_node_named(words[1]);
    node2 = find_node_named(words[2]);
}

// CHECK FOR ERRORS
if ((node1==null) || (node2==null))
{
    GUI.ERROR(0,"DataSea.word_unlink","Can't find either node1 or node2");
    return;
}

GUI.P(0,"DataSea.word_unlink","About to unlink '"+node1.Name+"' from '"+node2.Name+"'");
// NOW, DO WHAT'S ASKED OF US
node1.unlink_both(node2);

} // end word_unlink

/**
 ** word_link
 */

public void word_link (String[] words, int num_words) {
    Node node1=null, node2=null;

    GUI.P(0,"DataSea.word_link","Started.");

    /*****
    *****/

} // end word_link

/**
 ** word_most
 */

public void word_most (String[] words, int num_words) {
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

## DataSea.java

59

```
GUI.P(0,"DataSea.word_most","NO CODE YET.");  
// NOW, DO WHAT'S ASKED OF US
```

```
} // end word_most
```

```
/**
```

```
/**  
** word_rename  
*/
```

```
public void word_rename (String[] words, int num_words) {  
Node node=null;  
String new_name="DEFAULT_NEW_NAME";
```

```
GUI.P(0,"DataSea.word_rename","Started.");
```

```
// HANDLE IDENTIFYING THE CORRECT NODE TO START ON
```

```
if (num_words<2)
```

```
{  
GUI.ERROR(0,"DataSea.word_rename","Need at least one argument for new  
name.");
```

```
return;
```

```
if (num_words>3)
```

```
{  
GUI.ERROR(0,"DataSea.word_rename","Need at most two arguments: node a  
nd new_name.");
```

```
return;
```

```
if (num_words==2) {
```

```
node = gui.lastNode;  
new_name = words[1];
```

```
}
```

```
else
```

```
if (num_words==3) {
```

```
node = find_node_named(words[1]);  
new_name = words[2];
```

```
}
```

```
// CHECK FOR ERRORS
```

```
if (node==null)
```

```
{
```

```
GUI.ERROR(0,"DataSea.word_rename","Can't find node "+words[1]);
```

```
return;
```

```
}
```

```
// NOW, DO WHAT'S ASKED OF US
```

```
node.Name = new_name;  
} // end word_rename
```

```
/**
```

```
** and  
**
```

```
*/
```

```
public void and (String[] words, int num_words) {  
Node node_1, node_2;
```

```
int i;
```

```
node_1 = find_node_named(words[1]);
```

```
if (node_1==null)
```

```
{  
GUI.ERROR(0,"and","node_1 is null");  
return;
```

```
}
```

```
node_2 = find_node_named(words[2]);
```

```
if (node_2==null)
```

```
{  
GUI.ERROR(0,"and","node_2 is null");  
return;
```

```
}
```

```
GUI.P(0,"and","OK: node_1 is "+node_1.Name+"; node_2 is "+node_2.Name);
```

```
back_r((Node)node_1, node_2, 0);
```

```
} // end and
```

```
/**
```

```
** print_print_upstream  
**
```

```
*/
```

```
public void print_print_upstream (Node caller, Node node) {
```

```
int i, size;
```

```
Node child;
```

```
if (node == null) {
```

```
GUI.ERROR(0,"DataSea.print_print_upstream","NULL node given");  
return;
```

```
}
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

## DataSea.java

60

```
if (caller == null)
    System.out.println("");
else
    System.out.print("-> "+node.Name+"{" +node.dist+"}");

size = node.Links.size();
for (i=0; i<size; i++) {
    child = node.getNodeAtLink(i);
    if (node.goesUpstreamTo(child)) {
        if (caller == null)
            System.out.println(node.Name+"{" +node.dist+"}");
        print_upstream(node, child);
    }
    else
        System.out.println("");
}

} // end print_upstream

/**
 * method Back like back(node), but set_Tdist is run on all children of node
 */
public void Back (String[] words, int num_words) {
    int i, size;
    Node node, child, grand_child;

    if (num_words<1)
        return;
    if (POV==null)
        return;

    // HANDLE IDENTIFYING THE CORRECT NODE TO START ON
    if (num_words==1)
        node = gui.lastNode;
    else
        node = find_node_named(words[1]);

    if (words[0].equalsIgnoreCase("whats")) {
        whats = true;
    }

    // CHECK FOR ERRORS
    if (node==null) {
        if (num_words>1)
            GUI.WARNING(0, "DataSea Back", "Can't find node "+words[1]);
        else

```

```
GUI.WARNING(0, "DataSea Back", "Neither Name given nor existing lastNode.");
        return;
    }
    else
        GUI.P(1, "DataSea.Back", "Found node named '"+node.Name+"'");

    // ----- for children of POV child -----
    child = POV.getNodeAtLink(0);
    size = child.Links.size();
    for (i=0; i<size; i++) {
        grand_child = (Node)(child.getNodeAtLink(i));
        GUI.P(0, "DataSea.Back", "Working on POV child named '"+child.Name
            +"'" +grand_child named '"+grand_child.Name+"'");
        set_Tdist_start(grand_child); // ORDER FROM ALL CHILDREN OF POV CHILD, THE
        N RUN back_r()
        back_r((Node)null, node, 0);
    }
    // ----- end for children of POV child -----

} // end Back

/*
 *
 */
public Node figure_out_node (String caller_fn, String[] words, int num_words) {
    Node node;

    if (num_words<1)
        return((Node)null);
    // HANDLE IDENTIFYING THE CORRECT NODE TO START ON
    if (num_words==1)
        node = gui.lastNode;
    else
        node = find_node_named(words[1]);

    if (node==null) {
        GUI.WARNING(0, caller_fn, "Null node, quitting.");
    }
    else
        GUI.P(0, caller_fn, "Found node named '"+node.Name+"'");

    return(node);
} // end figure_out_node

/**

```

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin DataSea.java

61

```

** method back3
*/
    public void back3 (String[] words, int num_words) {
        Node node;

        if (null == (node = figure_out_node("back3", words, num_words)))
            return;

        set_Tdist.start(node); // NEED FOR SETTING VARIABLE 'TDIST'

        System.err.println("");
        back2(node);
        System.err.println("");
        return;
    } // end back3

/**
 ** back2
 **
 */
    public boolean back2 (Node node) {
        int i, size;
        boolean ret_val = false;
        Node child=null;

        size = node.Links.size();
        for (i=0; i<size; i++) {
            child = node.getNodeALink(i);

            if (child.dist == POV.dist+1) {
                ret_val = true;
                child.more_mag();
                System.out.println(++ back2; next to POV, "
                    +"mag++ of "+child.Name);
                return(ret_val);
            }

            if (child.dist < node.dist) { // we may do something
                //System.out.println(++ back2; at POV+1 point, mag++ node="+node.Name+", child="+child.
                Name);
                //System.out.println(" back2;"+node.Name+"-> recursing on "+child.Name);
                if (true==back2(child)) {
                    ret_val = true;
                    System.out.println(++ back2; path-point, child<node, "
                        +"mag++ of "+child.Name);
                }
            }

            child.more_mag(); // else ret_val is false
        }
    }

    //System.err.println(" returning("++ret_val+ "<"+node.Name+" ,Tdist="+node.dist+ ")");
    return(ret_val);
    //else if (child.Type==node.Type) {} // recurse if same type and not next to POV
} // end back2

/**
 ** method back amplify mag going backwards, calls back_r
 */
    public void back2 (String[] words, int num_words) {
        int i,j,k, size, size1, size2;
        Node node, child1=null, child2=null, child3=null;

        /*****
        if (null == (node = figure_out_node("back", words, num_words)))
            return;
        *****/
        if (null == (node=POV)) {
            System.out.println("back(): need a POV, ");
            return;
        }

        System.out.println("back(): starting on POV.");

        // 'node' is POV
        // ----- for children -----
        size = node.Links.size();
        for (j=0; j<size; j++) {
            child1 = (Node)(node.getNodeALink(j));
            System.out.println("child1 is "+child1.Name);
            size1 = child1.Links.size();
            for (i=0; i<size1; i++) {
                child2 = (Node)(child1.getNodeALink(i));
                if (child2 != node) {
                    System.out.println(" child2 is "+child2.Name);
                    //else
                    //System.out.println(" (skipping) "+child2.Name);
                }
                size2 = child2.Links.size();
                for (k=0; k<size2; k++) {
                    child3 = (Node)(child2.getNodeALink(k));
                }
            }
        }
    }

```

05/23/00  
00:48:20

62

```

        if (child3 != child1) {
            System.out.println("
            System.out.println("
            *)", isPolarized="+child3.isPolarized);
            //else
            //System.out.println("
            if (child3.isEvent)
                mag.downhill_event(child3);
        }
    }

    //-----end for children -----
    return;
} // end backt

/**
 ** mag.downhill_event
 **
 */
public void mag_downhill_event (Node node) {
    int i, size;
    Node child;

    if (node.isEvent)
        node.more_mag();
    size = node.Links.size();
    for (i=0; i<size; i++) {
        child = node.getNodeAtLink(i);
        System.out.println("Is "+child.Name+" downstream from '"+node.Name+"'? "+node.goesDownstreamTo(child));
        if (node.goesDownstreamTo(child))
            mag_downhill_event(child);
    }
} // end mag_downhill_event

/**
 ** mark_distally
 */
}

    **
    */
    public void mark_distally (Node node) {
        int i, size;
        Node child;

        node.isMarked = true;
        size = node.Links.size();
        for (i=0; i<size; i++) {
            child = (Node)node.getNodeAtLink(i);
            if (child.dist > node.dist)
                mark_distally(child);
        }
    }

    return;
} // end mark_distally

/**
 ** method showdist
 */
public void showdist (Node node, int Tdist) {
    Node child;
    int i, size;

    //-----for children -----
    size = node.Links.size();
    for (i=0; i<size; i++) {
        child = (Node)node.getNodeAtLink(i);
        if (child.Tdist == Tdist) {
            gui.dump_node(0, false, child);
            gui.show_node_once(child); // experiment
            gui.sleep(300);
        }
    }
    if (child.Tdist > node.Tdist) // recurse only if we go distal
        showdist(child, Tdist);
}

//-----end for children -----
return;
} // end showdist

/**
 ** method showdist
 */
public void showdist (String[] words, int num_words) {
    Node node=null;
}
```

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin

DataSea.java

63

```
int i, size, Tdist=0;

if (num_words == 1) {
    if (gui.lastNode != null)
        node = gui.lastNode;
    else
        GUI.WARNING(0, "showdist", "no name given, nor lastNode");
}
else if (num_words > 1) {
    node = find_node_named(words[1]);
    // CHECK FOR ERRORS
    if (node==null)
    {
        GUI.ERROR(0, "DataSea.showdist", "Can't find node "+words[1]);
        return;
    }
}
if (num_words > 2) {
    if (words[2].equals("1"))
        Tdist = 1;
    else
        if (words[2].equals("2"))
            Tdist = 2;
        else
            if (words[2].equals("3"))
                Tdist = 3;
            else
                if (words[2].equals("4"))
                    Tdist = 4;
                else
                    if (words[2].equals("5"))
                        Tdist = 5;
}
gui.getToolkit().sync(); // wait for things to calm down

System.err.println("showdist, Tdist set to "+Tdist);
set_Tdist_start(node);
showdist(node, Tdist);
return;
} // end showdist

/**
 ** method showdist

public void showdist (Node node) {
    int i;

    if (node == null)
        return;
    add_POV();
    POV.link(node);
    set_dist_start(POV); // need for use with set_Tdist, recursive recursion logic
    POV.setLinks_VRparmsTto(node); // USE THE EXISTING VR PARMS IN THE NEW
    LINK
    GUI.P0, "DataSea.showdist", "Linking node "+node.Name);
    mag(node, "both", "+", 2);
    gui.lastNode = node;
} // end showdist(Node node)
*/

/**
 ** method showCNs
 */
public void showCNs () {
    int i, size;
    Node node, tn=null, CNode=null;

    if (gui.lastNode == null)
        return;
    node = gui.lastNode;

    GUI.P0, "DataSea.showCNs", "Called on node "+node.Name);
    gui.drawCN = true;

    size = node.Links.size();
    for (i=0; i<size; i++) {
        tn = (Node)(node.getNodeAtLink(i));
        CNode = node.getCNodeAtLink(i);
        if (CNode != null) {
            GUI.P0, "DataSea.showCNs", "Found CNode "+CNode.Name);
            CNode.set_mag(Node.MAX_MAG);
        }
    }
    return;
} // end showCNs

/**
 ** method dumpCNs
```

05/23/00  
00:48:20

Confidential and Proprietary Property of Rocky Nevin  
DataSea.java

64

```
*/
public void dumpCNs () {
    int i, size;
    Node node, tn=null, CNode=null;

    // ----- for all nodes -----
    size = node_vec.size();
    for (i=0; i<size; i++) {
        tn = (Node)(node_vec.elementAt(i));
        if (tn.isCN)
            gui.dump_node(0, true, tn);
    }
    // ----- end for all nodes -----

    return;
} // end dumpCNs

/**
 ** freeze
 **
 */
public void freeze () {
    int i, size;
    Node child;

    size = node_vec.size();
    for (i=0; i<size; i++) {
        child = (Node)(node_vec.elementAt(i));
        child.isFrozen = true;
    }

} // end freeze

/**
 ** unfreeze_all
 **
 */
public void unfreeze_all () {
    int i, size;
    Node child;

    size = node_vec.size();
    for (i=0; i<size; i++) {
        child = (Node)(node_vec.elementAt(i));
        child.isFrozen = false;
    }

} // end unfreeze_all

    } // end unfreeze_all

    return;
} // end unfreeze_r

/**
 ** unfreeze_r Recursive version
 **
 */
public void unfreeze_r (Node node) {
    Node child;
    int i, size;
    node.isFrozen = false;

    size = node.links.size();
    for (i=0; i<size; i++) {
        child = node.getNodeAtLink(i);
        if (child.dist > node.dist)
            unfreeze_r(child);
    }

    return;
} // end unfreeze_r

/**
 ** method show
 **
```

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin

DataSea.java

05

```
*/
public void show (Node node) {
    int i;
        if (node == null)
            return;
        GUI.P(0,"DataSea.show","Called on node "+node.Name);
        add POV();
        POV.link(node);
        //set dist.start(POV); // need for use with set Tdist recursive recursion logic
        POV.setLinksVRparmsTo(node); // USE THE EXISTING VR PARMS IN THE NEW
LINK
        GUI.P(0,"DataSea.show","Linking node "+node.Name);
        node.set_mag(Node.EMPHASIZED_MAG);
        gui.lastNode = node;
        needdistUpdate = true;
    } // end show(Node node)

/**
** method show
*/
public void show (String input_string) {
    Node node;
    int num_words, jnum_words;
    String jwords[];

    StringTokenizer st = new StringTokenizer(input_string, "");
    num_words = st.countTokens();
    String[] words = new String[num_words];

    System.err.println("num_words is "+num_words+", input_string is <"+input_string+">");
    for(int i = 0; i < num_words; i++) {
        words[i] = st.nextToken();
        System.err.println("broken into words: <"+words[i]+>");
        if (i>0) {
            Node node;
            node = find_node_named(words[i]);
            if (node != null) {
                System.err.println("back_r on "+node.Name+", Tdist="+node.dist);
                back_r((Node)null, node, 0);
                normalize();
            }
        }
    }

    StringTokenizer st = new StringTokenizer(words[i], "<>\\|\\|\\|");
    jnum_words = st.countTokens();
    if (jnum_words > 0) {
        jwords = new String[jnum_words];
        for(int j = 0; j < jnum_words; j++) {
            jwords[j] = st.nextToken();
            if (i==0 && j==1) {
                show(find_node_named(words[j]));
            }
        }
    }
    return;
} // end show

    public void show2 (String[] words, int num_words) {
        Node node;

        if (null == (node = figure_out_node("show2", words, num_words)))
            return;
        show (node);
    } // end show2

/**
** method show alternate arguments
*/
    public void show2 (String word) {
        Node node;

        String s[] = new String[2];
        s[0] = new String ("show");
        s[1] = new String (word);
        show2(s, 2);
    } // end show2

/**
** method word_save
*/
    public void word_save () {
        gui.maescale = 1;
    }
}
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

## DataSea.java

66

```
absorb_POV(true);
reset_mags();
reset_selected();
gui.globalMaxPressure = 1.0;
local(Root);
simplify_recursive((Node)null, Root);
} // end word_save

/**
 ** method word_reset
 */
public void word_reset () {
    gui.global_str_size = 0;
    gui.magscale = 1;
    unfreeze_all();
    absorb_POV(false);
    reset_mags();
    reset_selected();
    gui.globalMaxPressure = 1.0;
    local(Root);
    simplify_recursive((Node)null, Root);
} // end word_reset

/**
 ** method reset_selected
 */
public void reset_selected () {
    int i, size;
    double x, y;
    Node tn;

    gui.global_str_size = 0;

    size = node_vec.size();
    GUI.P(2, "reset_selected", "Resetting selected");
    for (i=0; i<size; i++) {
        tn = (Node)(node_vec.elementAt(i));
        if (tn != null) {
            tn.isSelected = false;
            tn.isMarked = false;
            if (tn.isAN()) {
                tn.x = tn.X;
                tn.y = tn.Y;
            } else {
                x = 0;
                y = 0;
                tn.x = x;
            }
        }
    }
}

}

tn.y = y;

}

}

/**
 ** method word_zoom
 */
public void word_zoom (String[] words, int num_words) {
    if (num_words == 1) {
        if (gui.lastNode != null)
            zoom(gui.lastNode.Name);
        else
            GUI.WARNING(0, "word_zoom", "no name given, nor lastNode");
    }
    else if (num_words > 1) {
        zoom(words[1]);
    }
} // end word_zoom

public void zoom (String name) {
    Node node;
    double i, smooth_motion_frames=2, temp_XOffset=0, temp_YOffset=0;

    node = find_node_named(name);
    if (node != null) {
        temp_XOffset = (node.x - gui.Window.XOffset) / smooth_motion_frames;
        temp_YOffset = (node.y - gui.Window.YOffset) / smooth_motion_frames;
        for (i=0; i<smooth_motion_frames; i++) {
            gui.Window.XOffset += (int)temp_XOffset;
            gui.Window.YOffset += (int)temp_YOffset;
            gui.update(1);
        }
        GUI.P(1, "DataSea.zoom", "Centering on "+node.Name+" (x,y)=( "+node.x+", "+node.y+"
    )");
        GUI.P(1, "DataSea.zoom", "Window.XOffset="+gui.Window.XOffset);
        GUI.P(1, "DataSea.zoom", "Window.YOffset="+gui.Window.YOffset);
    }
    else
        GUI.WARNING(0, "DataSea.zoom", "Can't find node "+name+"");
}

}
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin DataSea.java

67

```
    }  
    } // end zoom  
  
    /**  
     * method word_whats  
     */  
    public void whats (String[] words, int num_words) {  
        String word1=null, word2=null, AnswerStr;  
        Node node1, node2, child;  
        int index, size, i;  
  
        if (num_words<3) {  
            GUI.WARNING(0,"whats", "Need three words, returning.");  
            return;  
        }  
  
        if (0 < (index = words[1].indexOf("s")) {  
            word1 = words[1].substring(0,index); // whats BOB S hair  
            node1 = find_node.named(word1);  
        }  
        else {  
            GUI.WARNING(0,"whats", "No possessive");  
            return;  
        }  
  
        word2 = words[2];  
        node2 = find_node.named(word2);  
        GUI.P(0,"whats", "\Using word1="+word1+" , word2="+word2+"["+words[2]+"["+node2.Name+""]");  
        if (node1==null) {  
            GUI.WARNING(0,"whats", "node1 is null, returning.");  
            return;  
        }  
        if (node2==null) {  
            GUI.WARNING(0,"whats", "node2 is null, returning.");  
            return;  
        }  
    }  
  
    // So, now we have two valid nodes. Find node between them.  
  
    size = node1.links.size();  
    for (i=0; i<size; i++) {  
        child = (Node)(node1.getNodalLink(i));  
  
        if (child == node2.getParent()) {  
            GUI.P(0,"whats", AnswerStr="What's "+words[1]+" "+words[2]+" :");  
            gui.global_str[gui.global_str.size=0] = AnswerStr;  
            AnswerStr=" ANSWER: --> "+child.Name+" is "+words[1]+" "+words[2]+" :";  
            GUI.P(0,"whats", AnswerStr);  
            gui.global_str[gui.global_str.size=1] = AnswerStr;  
            gui.global_str[gui.global_str.size=2] = "=====  
            }  
        }  
        } // end whats  
  
        /**  
         * store string into global_str  
         */  
        public void store_string_into_global_str (String str) {  
            gui.global_str[gui.global_str.size++] = str;  
        } // end store_string_into_global_str  
  
        /**  
         * method word_trace  
         */  
        public void word_trace (String[] words, int num_words) {  
            Node node1, node2;  
            Node child;  
            int i;  
  
            if (num_words==1)  
                node1 = gui.SavedNode;  
            else  
                node1 = find_node.named(words[1]);  
            if (node1 == null) {  
                GUI.WARNING(0,"word_trace", "No node given or saved for node1");  
                return;  
            }  
            node2 = gui.lastNode;  
            if (node2==null) {  
                GUI.WARNING(0,"word_trace", "Need a lastNode");  
                return;  
            }  
        }  
    }  
}
```

```
GUI.P(0,"word,trace", "Tracing from <"+node1.Name+"> to <"+node2.Name+">");

boolean Saved_StopsSpread = model.StopsSpread; // Temporarily change it
model.StopsSpread = true;

store_string_into_global_str("Explaining why <"+node2.Name+"> is enhanced.");
set_dist_start(node2); // set dist from node2, trace back from node1
trace(model, node2);
set_dist_start(POV); // restore dist's

model.StopsSpread = Saved_StopsSpread; // Restore it
return;
} // end word,trace

/**
 ** trace call traceback on all children with path to POV
 **
 */
public void trace (Node model, Node node2) {
    int i, size;
    Node child;
    Link link=null;
    String str="";

    if (model==null) {
        gui.WARNING(0,"trace", "model is null.");
        return;
    }
    if (node2==null) {
        gui.WARNING(0,"trace", "node2 is null.");
        return;
    }
    // model.set_mag(Node.MAX_MAG);
    // model.isSelected = true;

    // ----- for children -----
    size = model.Links.size();
    for (i=0; i<size; i++) {
        child = (Node)(model.getNodeAtLink(i));
        if (child.dist != -1) {
            if (child.hasSmallerDistThan(model) && child.mag>=Node.MED_MAG) {
                link = model.getLink(i);

                gui.p(str="trace: node (" +model.Name+") has link named [" +link.Name+"] ...");
                store_string_into_global_str(str);
                gui.p(str="trace: ... to node (" +child.Name+")");
                store_string_into_global_str(str);
                trace(child, node2);
            }
        }
        // ----- end for children -----
    }

    } // end trace
    /*
    **
    **/
    public void traceback (Node model, Node node2) { // mag and select paths back to POV
        Node parent;
        String str;
        int i, size;
        Node child;
        Link link=null;

        // store_string_into_global_str(str);
        //gui_global_str[gui_global_str_size++] = "Working on Node="+model.Name;

        if (model == null)
            return;
        model.set_mag(Node.EMPHASIZED_MAG);

        // ----- for children -----
        size = model.Links.size();
        for (i=0; i<size; i++) {
            child = (Node)(model.getNodeAtLink(i));
            if (child.dist != -1) { // That is, there is a path to POV from here ...
                if (child.dist < (model.dist-0.0001)) {
                    link = model.getLink(i);
                    System.out.println(" <"+model.Name+"> (" +model.dist+" ) -----> "+link.Name+
                        "-----> "+child.Name+ "> (" +child.dist+" )");
                    traceback(child, node2);
                }
            }
        }
    }
    return;
}
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

## DataSea.java

69

```
} // end traceback

/**
 ** method parse_set_cmd set's the mag of a node
 */
public double parse_set_cmd (String[] words, int num_words) {
    double val = 0;

    if (gui.lastNode == null) {
        GUI.P(0,"parse_set_cmd",
            "No lastNode, value gotten is "+GUI.java_lang_double.valueOf(words[2]));
    }

    if (num_words != 3) {
        GUI.P(0,"parse_set_cmd","num_words wrong, "+num_words);
        return(val);
    }
    else {
        val = (double)(GUI.java_lang_double.valueOf(words[2])).doubleValue();
        if (words[1].equals("mag"))
            gui.lastNode.set_mag(val);
        if (words[1].equals("importance"))
            gui.lastNode.importance = val;
    }

    // GUI.java_lang_double.valueOf(words[2]);
    GUI.I(0,"parse_set_cmd", "" + ((double)1.0 + (GUI.java_lang_double.valueOf(words[2])).doubleValue());
    return(val);
} // end parse_set_cmd

/**
 ** print_triplets Print information based on triplet-configurations
 **
 */
public void print_triplets (Node node) {
    int i, size, j, jsize;
    Node child=null, grand_child=null;
    String str; // to put a line of information into
    String desc; // to get or set as Desc of link from AN to DN

    gui.global_str_size = -1; // reset this

    size = node.Links.size();
    for (i=0; i<size; i++) {
        child = (Node)(node.getNodeAtLink(i));

        jsize = child.Links.size();
        for (j=0; j<jsize; j++) {
            grand_child = (Node)(child.getNodeAtLink(j));
            if (grand_child.isAN && child.isDN && child.mag >= Node.MED_MAG) {
                desc = grand_child.get_Desc(child);
                if (desc.equals(""))
                    desc = "is";
                str = "("+gui.prec(child.mag, 3)+") "
                    +node.Name+"s "+grand_child.Name+" "
                    +desc+" "+child.Name;
                gui.global_strf++gui.global_str_size = str;
                GUI.P(0,"print_triplets", gui.global_strf[gui.global_str_size]);
            }
        }
    }
    return;
} // end print_triplets

/**
 ** print_blanks
 **
 */
public void print_blanks (int number_of_blanks) {
    int i;

    for (i=0; i<number_of_blanks; i++) {
        System.out.print(" ");
    }
    return;
} // end print_blanks

/**
 ** print
 **
 */
public void print () {
    int i, size;
    Link link=null;
}
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

DataSea.java

70

```
String s="";
Node child;

if (POV == null)
    return;

size = POV.links.size();
for (i=0; i<size; i++) {
    child = POV.getNodeALink(i);
    if (child.isAN && !gui.drawAN)
        ;
    else {
        link = POV.getLink(i);
        s = "→["+child.Name+" (" +child.mag+")("+link.Name+")]";
        s = s + "
";
        System.out.print(s.substring(0,20));
        print_r(POV, child, 1);
    }
}
return;
} // end print

/**
 ** print_r
 **
 */
public void print_r (Node parent, Node child, int level) {
    int i, size;
    Node grand_child;
    Link link=null;
    int count=0;
    String s;

    if (parent == null)
        return;
    if (child == null)
        return;

    size = child.links.size();
    for (i=0; i<size; i++) {
        grand_child = child.getNodeALink(i);
        if ((grand_child != parent)
            && (grand_child.dist > child.dist)) {

            if (child.isAN && !gui.drawAN)
                ;
            else {
                if (grand_child.mag > Node.BIG_MAG)
                    {
                        if (++count > 1)
                            print_blanks(20*level);
                        link = child.getLink(i);
                        s = " [" +grand_child.Name+" (" +grand_child.mag+")("+link.Name+")]";
                        s = s + "
";
                        if (child.mag > Node.BIG_MAG) { // See if the prior node is big also
                            System.out.print(s.substring(0,20));
                            print_r(child, grand_child, level+1); // Recurse w/ increment
                        }
                    }
                else {
                    System.out.print("....." + s.substring(0,20));
                    print_r(child, grand_child, level); // Recurse w/o increment
                }
            }
        }
    }
    if (count == 0)
        System.out.println(""); // Only carriage return when no valid children exist
    return;
} // end print_r

/**
 ** method print
 */
public void word_print (String[] words, int num_words) {
    int i, size, max_dist=4;
    Node node, tn, saved_node=null;
    double saved_mag=0;

    /*****
    node = gui.lastNode;
    *****/
    if (num_words==1)
        node = gui.lastNode;
}
```

05/23/00  
00:48:20

Confidential and Proprietary Property of Rocky Nevin  
DataSea.java

74

```
else
    node = find_node_named(words[1]);
    if (node == null) {
        GUI.WARNING(0, "DataSea.print", "No node to start on.");
        return;
    }
    // So, continue ...

// PRINT TRIPLETS
print_triplets(node);

if (GlobalVec == null)
    GlobalVec = new Vector();
else
    GlobalVec.removeAlIElements();

store_ANs(r(node, max_dist, 0); // this puts all the ANs within dist=max_dist into the GlobalVec

size = GlobalVec.size();
for (i=0; i<size; i++) {
    tn = (Node)(GlobalVec.elementAt(i));
    if (tn.mag > saved_mag) {
        if (saved_node != null)
            saved_mag = tn.mag;
            saved_node = tn;
        }
    }

// now, saved_mag and _node have the biggest AN within max_dist
if (saved_node==null) {
    GUI.WARNING(0, "word_print", "Didn't process mag's well, saved_node == null");
    return;
}
else
    GUI.P(0, "words input", "Max AN mag of "+saved_node.Name+" is "+saved_node.mag);

// gui_global_str_size = -1; used by print_triplets also
System.err.println("=====");
System.err.println("Predominant Categories from "+saved_node.Name);
gui_global_strf++gui_global_str_size] = "Predominant Categories from "+saved_node.Name;
size = saved_node.Links.size();
for (i=0; i<size; i++) {
    tn = (Node)(saved_node.getNodeAtLink(i));
    if (tn.isAN && tn.dist <= saved_node.dist && tn.dist > gui_lastNode.dist) {
        System.err.println("-----"+saved_node.Name);
        gui_global_strf++gui_global_str_size] = "-----";
    }
}

"+saved_node.Name;
go_backwards.r(n);
System.err.println("");
}

if (POV != null) {
    // Look at high-mag children and grandchildren of current POV
    // ----- for children of POV -----
    int i_size, j_size, j;
    Node child=null, grand_child=null;

    gui_global_strf++gui_global_str_size] = "POV Children and Grandchildren.";

    i_size = POV.Links.size();
    for (i=0; i<i_size; i++) {
        child = (Node)(POV.getNodeAtLink(i));
        if (child.mag >= Node.BIG_MAG)
            gui_global_strf++gui_global_str_size] = " "+child.Name+"("+child.mag+")";
        j_size = child.Links.size();
        for (j=0; j<j_size; j++) {
            grand_child = (Node)(child.getNodeAtLink(j));
            if (grand_child.mag >= Node.BIG_MAG)
                gui_global_strf++gui_global_str_size] = " "+grand_child.Name+"("
            )
        }
    }
    // ----- end for children of POV -----
}

System.out.println("TEST....print_upstream().....");
print_upstream((Node)null, node);
System.out.println("TEST....print_upstream() done.....");

System.err.println("=====");
return;
} // end print

/**
 ** go_backwards
 **
 */
public void go_backwards r (Node node) {
```

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

DataSea.java

7/2

```
int i, size;  
Node tn;
```

```
System.err.print("-> "+node.Name);
```

```
gui.global_str[gui.global_str.size] = gui.global_str[gui.global_str.size]+" -> "+node.Name;
```

```
size = node.Links.size();
```

```
for (i=0; i<size; i++) {
```

```
    tn = (Node)(node.getNodeAtLink(i));
```

```
    if (tn.isAN && tn.dist < node.dist && tn.dist > gui.lastNode.dist)
```

```
        go_backwards_r(tn);
```

```
}
```

```
return;
```

```
} // end go_backwards_r
```

```
/**
```

```
 ** store_ANs
```

```
 **
```

```
 */
```

```
public void store_ANs_r (Node node, int max_dist, int this_dist) {
```

```
    int i, j, size, isize;
```

```
    Node tn;
```

```
    boolean not_found=true;
```

```
    this_dist++;
```

```
    size = node.Links.size();
```

```
    for (i=0; i<size; i++) {
```

```
        tn = (Node)(node.getNodeAtLink(i));
```

```
        not_found = true;
```

```
        if (tn.isAN) {
```

```
            if (GlobalVec.contains(tn))
```

```
                not_found = false;
```

```
        } //*****
```

```
        isize = GlobalVec.size(); // See if we have it already
```

```
        for (j=0; j<isize; j++) {
```

```
            if ((Node)(GlobalVec.elementAt(j)) == tn)
```

```
                not_found = false;
```

```
        } //*****
```

```
        } //*****
```

```
        if (not_found)
```

```
            {  
                GlobalVec.addElement(tn);
```

```
            }
```

```
        if ((this_dist <= max_dist) && node.goesUpstreamTo(tn)) // recurse
```

```
            store_ANs_r(tn, max_dist, this_dist);
```

```
    }
```

```
} // end store_ANs
```

```
/**
```

```
 ** method word_dump
```

```
 */
```

```
public void word_dump (String[] words, int num_words) {  
    Node node;
```

```
    if (num_words==0) {
```

```
        GUI.WARNING(0, "DataSea.word_dump", "No words received at all.");
```

```
        return;
```

```
    }
```

```
    if (num_words==1)
```

```
        GUI.P(1, "DataSea.word_dump", "One word received '"+words[0]+"'");
```

```
    if (num_words==2)
```

```
        GUI.P(1, "DataSea.word_dump", "Two words received '"+words[0]+"', '"+words[1]+"';
```

```
    );
```

```
    if (num_words==3)
```

```
        GUI.P(1, "DataSea.word_dump", "Three words received '"+words[0]+"', '"+words[1]+"',
```

```
        '"+words[2]+"';");
```

```
    if (num_words==1)
```

```
        node = gui.lastNode;
```

```
    else
```

```
        node = find_node_named(words[1]);
```

```
    if (node != null) {
```

```
        if (words[0].equals("d"))
```

```
            gui.dump_node(0, false, node);
```

```
        else
```

```
            gui.dump_node(0, true, node);
```

```
    }
```

```
} // end word_dump
```

```
/**
```

```
 ** method undo
```

```
 */
```

```
public void undo (int levels) {
```

```
    int i, size;
```

```
    Node tnode;
```

```
    GUI.P(1, "undo", "Begun, "+levels+" levels.");
```

```
    size = node_vec.size();
```

```
    for (i=0; i<size; i++)
```

```
    {
```

```
        tnode = (Node)(node_vec.elementAt(i));
```

```

    /**
     ** parse_event_input
     **
     */

    public double parse_event_input(String s) {
        double offset = 0;

        offset = 0;

        if (s.equalsIgnoreCase("tomorrow"))
            offset = .01; // was .66

        if (s.equalsIgnoreCase("today"))
            offset = .01; // was .33

        if (s.equalsIgnoreCase("yesterday"))
            offset = .01;

        if (s.equalsIgnoreCase("noon"))
            offset = (1.0/24.0)*12.0 * 0.33;

        if (s.equalsIgnoreCase("1pm"))
            offset = (1.0/24.0)*13.0 * 0.33;

        if (s.equalsIgnoreCase("2pm"))
            offset = (1.0/24.0)*14.0 * 0.33;
    }
}

```

-269-

05/23/00  
00:48:20

# Confidential and Proprietary Property of Rocky Nevin

DataSea.java

74

```
if (node == null)
    return;

GUI.P(0,"TS","TimeStamping "+node.Name);

if (null == (created = find_node_named("Created", "AN"))) {
    GUI.P(0,"TS","Creating node 'Created'");
    created = pop.create_node("Created", "AN");
    created.link(Root);
}

node = pop.create_node(""+GUI.current_TS, "Event"); // creates the event
node.link(created); // link to "Created"
node.link(pop.Timeline); // link to Timeline

return;
} // end TS

/**
 ** group assumes the mag of distal ANs has been additively increased to
 ** emphasize ANs of more relevance to distal DNs.
 ** Only group at the level 'target_level'.
 */
public void group (Node node, int target_level) {
    int i, size;
    Node inode;

    if (node == POV) {
        gui.pressure_mag /= 3;
    }

    if (node == null)
    {
        GUI.ERROR(0,"group", "NULL node");
        return;
    }

    GUI.P(0,"group", "level "+target_level);

    size = node.links.size();

    if (node.dist < target_level) // we aren't there yet
        for (i=0; i<size; i++)
        {
            inode = node.getNodeAtLink(i);
            if (inode.dist > node.dist // was >=, got infinite recursion w/ 2 shows

```

05/23/00  
00:48:20

## Confidential and Proprietary Property of Rocky Nevin

DataSea.java

7/5

```
} // end group_onto_biggest_AN

/**
 ** link_DN_to_ANS
 **
 * public void link_DN_to_ANS(Node dn_node, Node an_node) {
 *   int i, size;
 *   Node node;
 *
 *   if (an_node == null)
 *   {
 *     size = dn_node.Links.size();
 *     GUI.P(0, "link_DN_to_ANS", " NULL an_node (start), size of dn_node is "+size
 * );
 *     for (i=0; i<size; i++)
 *     {
 *       node = dn_node.getNodeAtLink(i);
 *       link_DN_to_ANS(dn_node, node);
 *       return;
 *     }
 *   }
 *   else if (!an_node.isAN)
 *   {
 *     GUI.P(0, "link_DN_to_ANS", " an_node is not Type AN");
 *     return;
 *   }
 *   else
 *   {
 *     {
 *       dn_node.link(an_node); // link() makes sure its not redundant
 *       size = an_node.Links.size();
 *       GUI.P(0, "link_DN_to_ANS", " recursing, linked "+an_node.Name+" to "
 * +dn_node.Name+" , size of "+an_node.Name+" is "+size);
 *       for (i=0; i<size; i++)
 *       {
 *         {
 *           (node = an_node.getNodeAtLink(i);
 *           link_DN_to_ANS(an_node, node);
 *           return;
 *         }
 *       }
 *     }
 *   }
 * } // end link_DN_to_ANS
 */

/**
 ** app_email
 **
 */
public void app_email () {
  int i;
  Node app_node, to_string_node, from_string_node, to_node, from_node, emailform_text_node;
  to_node = getNodeInNeighborhood("email");
  from_node = pop_create_node("rocky@allis.com", "DN", "email address for Rocky Nev
n");
  if (to_node == null) {
    GUI.WARNING(0, "app_email", "getNodeInNeighborhood didn't return a goo
d node");
    return;
  }
  app_node = new Node("EMF", "Form", "Email Form", 100, 100, 40, 80);
  emailform_text_node = new Node("text", "DN", "Email text", 0, 0, 5, 1);
  to_string_node = new Node("To:", "DN", "", 0, 0, 5, 1);
  from_string_node = new Node("From:", "DN", "", 0, 0, 5, 1);
  Root.link(app_node);
  app_node.link(to_string_node);
  app_node.link(from_string_node);
  app_node.link(emailform_text_node);
  app_node.link(to_node);
  app_node.link(from_node);
  // to_string_node.link(to_node);
  // from_string_node.link(from_node);
  // emailform_text_node.link(app_node); done above
  // the Y offset starts at the bottom and goes up, as y does in data space
  set_child_position(app_node, to_string_node, 0.01, 0.8);
  set_child_position(app_node, from_string_node, 0.01, 0.9);
  set_child_position(app_node, to_node, 0.4, 0.8);
  set_child_position(app_node, from_node, 0.4, 0.9);
  // set_child_position(to_string_node, to_node, 1.4, 1);
  // set_child_position(from_string_node, from_node, 2.0, 1);
  set_child_position(app_node, emailform_text_node, 0.01, 0.5);
  GUI.P(1, "app_email", "Done, select EmailForm now, in VR mode");
  needdistUpdate = true;
  reset_and_zoom("EMF");
  return;
} // end app_email
```

05/23/00  
00:48:20

76

```
/**
** reset_and_zoom
**
*/

public void reset_and_zoom (String target_name) {
    Node target_node;

    target_node = find_node_named(target_name);
    if (target_node == null) {
        GUI.WARNING(0, "reset_and_zoom", "can't find target node named "+target_name);
        return;
    }
    needdiUpdate = true;
    absorb_POV(false); // This stops the thread via gui.StopThreadRequest
    gui.mode_obj.set_render_mode("VR");
    mag(target_node, "both", "+");

    return;
} // end reset_and_zoom

/**
** get_node_in_neighborhood
**
*/

public Node get_node_in_neighborhood (String target_name) {
    Node ret_node = null;
    int size = 0;
    int i;

    GUI.PR(0, "get_node_in_neighborhood", "Looking for "+target_name);

// CHECK REFERENCE POINT
    if (gui.lastNode == null) {
        GUI.WARNING(1, "get_node_in_neighborhood", "gui.lastNode is null");
        return(ret_node);
    }

    target_node = find_node_named(target_name);

// CHECK TARGET POINT
    if (target_node == null) {
        GUI.WARNING(0, "get_node_in_neighborhood", "target_node is null");
        return((Node)null);
    }

    set_Tdist_start(gui.lastNode); // NEED FOR SETTING VARIABLE "TDIST"
    size = target_node.Links.size();
    for (i=0; i<size; i++) {
        in = target_node.get_node_at_link(i);
        GUI.PR(1, "get_node_in_neighborhood", "i is "+i+", in="+in.Name+", Type="+in.Type+", Tdist="+in.Tdist);
        if (((in.isAN) && (in.dist==target_node.dist-1)) {
            ret_node = in; // Got it
            GUI.PR(0, "get_node_in_neighborhood", "Got it: "+ret_node.Name);
        }
    }
    return(ret_node);
} // end get_node_in_neighborhood

/**
** set_child_position
**
*/

public void set_child_position (Node parent, Node child, int offsetX, int offsetY) {
    double theta, delta_x, delta_y;
    Link link;

    if (parent == null) {
        GUI.ERROR(0, "set_child_position", "parent is null");
        return;
    }
    if (child == null) {
        GUI.ERROR(0, "set_child_position", "child is null");
        return;
    }
    child.setX("set_child_position, parent="+parent.Name,
        offsetX,
        offsetY);
    link = parent.get_link_to(child);
    link.set_links_VParams(child);
} // end set_child_position
```

05/23/00  
00:48:20

Confidential and Proprietary Property of Rocky Nevin  
DataSea.java

77

```
/**
 ** set_child_position
 **
 */
public void set_child_position (Node parent, Node child, double fraction_width, double fr
action_height) {
    double theta, delta_x, delta_y;
    Link link;

    if (parent == null) {
        GUI.ERROR(0, "set_child_position", "parent is null");
        return;
    }
    if (child == null) {
        GUI.ERROR(0, "set_child_position", "child is null");
        return;
    }

    // child.setX("set_child_position, parent="+parent.Name,
    // (parent.size_X * fraction_width),
    // (parent.size_Y * (fraction_height-1)));

    delta_x = fraction_width - 0.5;
    delta_y = fraction_height - 0.5;

    theta = gui.get_angle(delta_x, delta_y);
    child.set_theta_offset(theta, "Called by set_child_position");
    link = parent.getLinkTo(child);
    link.setLinksVRparams(child);

    } // end set_child_position

} // End of class DataSea
```

05/24/99  
15:48:16

Confidential and Proprietary Property of Rocky Nevin  
LinkObj.java

1

```
import java.lang.*;

/*
 * This is Link.java by Rocky Nevin
 * @version 0.4.3/1/2/98
 */

public class LinkObj extends Object {
    String Name;
    String Type;
    String Description;
    double CS;
    Node Node;
    Node NodeL, NodeR;
    LinkObj NextLink;
    long Link.ID;

    /**
     * LinkObj CONSTRUCTORS
     */

    public LinkObj() {
        CS = 1;
        // RECURSES this.Node = new Node();
    }

    // End of class LinkObj
}
```

05/24/99  
15:48:33

Confidential and Proprietary Property of Rocky Nevin

VRObj.java

```
/**
 *
 * This is VRObj.java by Rocky Nevin
 *
 * <pre>
 * </pre>
 * @version 0.1. 8/9/98 Begun, based on prior program, G.java for DataSca
 * @author Rocky Nevin
 */
import java.lang.*;

//
// VR object holds info about the visual appearance on the screen,
// as part of a Node object

public class VRObj extends Object {

    double x, y, z;
    double dx=5, dy=5, dz=5; // Size of object
    double px, py, pz; // Pressure forces to move nodes
    double theta; // The angle from a horizontal line, left to right
    double radius; // The radius of this obj's sphere of influence
    String Appearance;

/**
 ** VRObj CONSTRUCTORS
 **
 */

    public VRObj () {

}

} // End of class VRObj
```

05/24/99  
16:11:37

# Confidential and Proprietary Property of Rocky Nevin

nsr.java

```

/*
 * nsr_position_node by Rocky Nevin
 */
public boolean nsr_position_node (Node parent, Node node) { //
boolean local_recurse=true;

if (parent==null) {
P3, "nsr_position_recursive", "NULL parent to node "+node.Name);
return(false);
}

P3, "nsr_position_node", "Starting on "+parent.Name+" -> "+node.Name);
if (node.Type.equals("FAB")) {
local_recurse=nsr_position_FAB(parent, node);
if (node.VR_node != VR_FAB.NODE)
P3, "nsr_position_node", "WRONG VR_FAB connection");
}
else if (node.Type.equals("MACH")) {
local_recurse=nsr_position_MACH(parent, node);
}
else if (node.Type.equals("DIR")) {
local_recurse=nsr_position_DIR(parent, node);
}
else if (node.Type.equals("DIR_ENTRY")) {
local_recurse=nsr_position_DIR_ENTRY(parent, node);
}
else if (node.Type.equals("TIMELINE")) {
local_recurse=nsr_position_TIMELINE(parent, node);
}
else {
node.x=parent.x + 50 + node.X;
node.y=parent.y + 50 + node.ChildIDNum + node.Y;
node.z=parent.z;
return(local_recurse);
} // end nsr_position_node
}

/**
 * method nsr_position_FAB a NSR function
 */
boolean nsr_position_FAB (Node caller, Node node) {
node.x=caller.x+50;
node.y=caller.y;
node.z=caller.z;
P3, "position_FAB", "Starting on "+caller.Name+" -> "+node.Name);
return(true);
} // end position_FAB

/**
 * method nsr_position_MACH a NSR function
 */
boolean nsr_position_MACH (Node caller, Node node) {
node.x+=0.5*(caller.x+node.X-node.x);
node.y+=0.5*(caller.y+node.Y-node.y);
node.z+=0.5*(caller.z+node.Z-node.z);
P3, "nsr_position_MACH", "Starting on "+caller.Name+" -> "+node.Name);
return(true);
} // end nsr_position_MACH

/**
 * method nsr_position_DIR a NSR function
 */
boolean nsr_position_DIR (Node caller, Node node) {
int i, size;
Node node;
P3, "position_DIR", "Starting on "+caller.Name+" -> "+node.Name);
node.x=caller.x+50;
node.y=caller.y;
node.z=caller.z;
size = node.Links.size();
for (i=0; i<size; i++) { // check dist of all children position if appropriate
node = node.getNodeAtLink(i);
if (node.dist == 1+node.dist)
nsr_position_DIR_ENTRY(node, node);
}
return(true);
} // end position_DIR

/**
 * method nsr_position_DIR_ENTRY a NSR function
 */
boolean nsr_position_DIR_ENTRY (Node caller, Node node) {
String Name;
int column,num=0;
Name=node.Name;
// CORRECT THIS PART
if (Name.equals("Name"))
column,num=1;
else if (Name.equals("Phone"))
column,num=2;
else if (Name.equals("Address"))
column,num=3;
}

```

05/24/99  
16:11:37

NSR.java

2

```

node.x=caller.x+50*column,num;
node.y=caller.y;
node.z=caller.z;

P(3,"position DIR_ENTRY",Starting on "+caller.Name+"->"+node.Name);

return(true);
} // end position_DIR_ENTRY

/**
 * method nsr_render_FAB a NSR function
 */
boolean nsr_render_FAB (Node caller, Node node) {
    Point ChildPoint=new Point();
    int width, height;

    map(node.x, node.y, ChildPoint);
    width=(int)(node.dx*magscale*node.mag);
    height=(int)(node.dy*magscale*node.mag);
    graphics.drawRect(ChildPoint.x, ChildPoint.y, width, height);
    graphics.drawString("FAB", ChildPoint.x, ChildPoint.y);

    P(3,"nsr_render_FAB",node.Name+"("+node.x+","+node.y+")");
    return(true);
} // end nsr_render_FAB

/**
 * method nsr_render_TIMELINE a NSR function
 */
boolean nsr_render_TIMELINE (Node caller, Node node) {
    Node node;
    Point ChildPoint=new Point();
    int width, height;

    map(node.x, node.y, ChildPoint);
    width=(int)(node.dx*magscale*node.mag);
    height=(int)(node.dy*magscale*node.mag);
    P(3,"nsr_render_TIMELINE",node.Name+"("+node.x+","+node.y+")");
    graphics.drawRect(ChildPoint.x, ChildPoint.y, width, height);
    graphics.drawString(node.Name, ChildPoint.x, ChildPoint.y);
    return(true);
} // end render_TIMELINE

/**
 * method nsr_render_MACH a NSR function
 */
boolean nsr_render_MACH (Node caller, Node node) {
    Node node;
    Point ChildPoint=new Point();
    int width, height;

    map(node.x, node.y, ChildPoint);
    width=(int)(node.dx*magscale*node.mag);
    height=(int)(node.dy*magscale*node.mag);
    P(3,"nsr_render_MACH",node.Name+"("+node.x+","+node.y+")");
    graphics.drawRect(ChildPoint.x, ChildPoint.y, width, height);
    return(true);
} // end render_MACH

graphics.drawRect(ChildPoint.x, ChildPoint.y, width, height);
graphics.drawString(node.Name, ChildPoint.x, ChildPoint.y);
return(true);
} // end render_MACH

/**
 * method nsr_render_DIR a NSR function
 */
boolean nsr_render_DIR (Node caller, Node node) {
    Point ChildPoint=new Point();
    int width, height;

    map(node.x, node.y, ChildPoint);
    width=(int)(node.dx*magscale*node.mag);
    height=(int)(node.dy*magscale*node.mag);
    graphics.drawRect(ChildPoint.x, ChildPoint.y, width, height);

    P(3,"nsr_render_DIR",node.Name+"("+node.x+","+node.y+")");
    return(true);
} // end nsr_render_DIR

/**
 * method nsr_render_DIR_ENTRY a NSR function
 */
boolean nsr_render_DIR_ENTRY (Node caller, Node node) {
    Point ChildPoint=new Point();
    int width, height;

    map(node.x, node.y, ChildPoint);
    width=(int)(node.dx*magscale*node.mag);
    height=(int)(node.dy*magscale*node.mag);
    graphics.drawRect(ChildPoint.x, ChildPoint.y, width, height);

    P(3,"nsr_render_DIR_ENTRY",node.Name+"("+node.x+","+node.y+")");
    return(true);
} // end nsr_render_DIR_ENTRY

```